

SESSION 8

iOS DEVELOPMENT

THE MANY WAYS TO

PERSIST DATA

WHAT IS DATA PERSISTENCE?

- ▶ Data stored in memory will be released when the user quits your app.
 - ▶ *Example: an array of objects on a view controller*
- ▶ Most apps need to **persist** (save) data so that users can leave and come back without losing information.

OPTIONS FOR DATA PERSISTENCE

- ▶ Store data on the device:

- ▶ User Defaults
- ▶ File Manager



We'll cover these two!

- ▶ Core Data

OPTIONS FOR DATA PERSISTENCE

- ▶ Store data remotely:

- ▶ Your own server

- ▶ Someone else's server



- ▶ iCloud

- ▶ Firebase

DATA PERSISTENCE

Done

USER DEFAULTS

- ▶ Intended to store small amounts of data
- ▶ Example: order of episodes

Podcasts

Settings

SHOW

Subscribed



Notifications



When new episodes are available they will be added to your library.

EPISODES

Play in Sequential Order

Play Most Recent First



Only Keep the Most Recent Episodes

Custom Settings

After the most recent episode is played, you'll return to older episodes.

Episodes will not be automatically downloaded but will be deleted when played based on your Podcasts Settings.

USER DEFAULTS

- ▶ Typically used to restore an app to the user's preferred default state
- ▶ But... you can store any values here

Podcasts

Settings

SHOW

Subscribed



Notifications



When new episodes are available they will be added to your library.

EPISODES

Play in Sequential Order

Play Most Recent First



Only Keep the Most Recent Episodes

Custom Settings

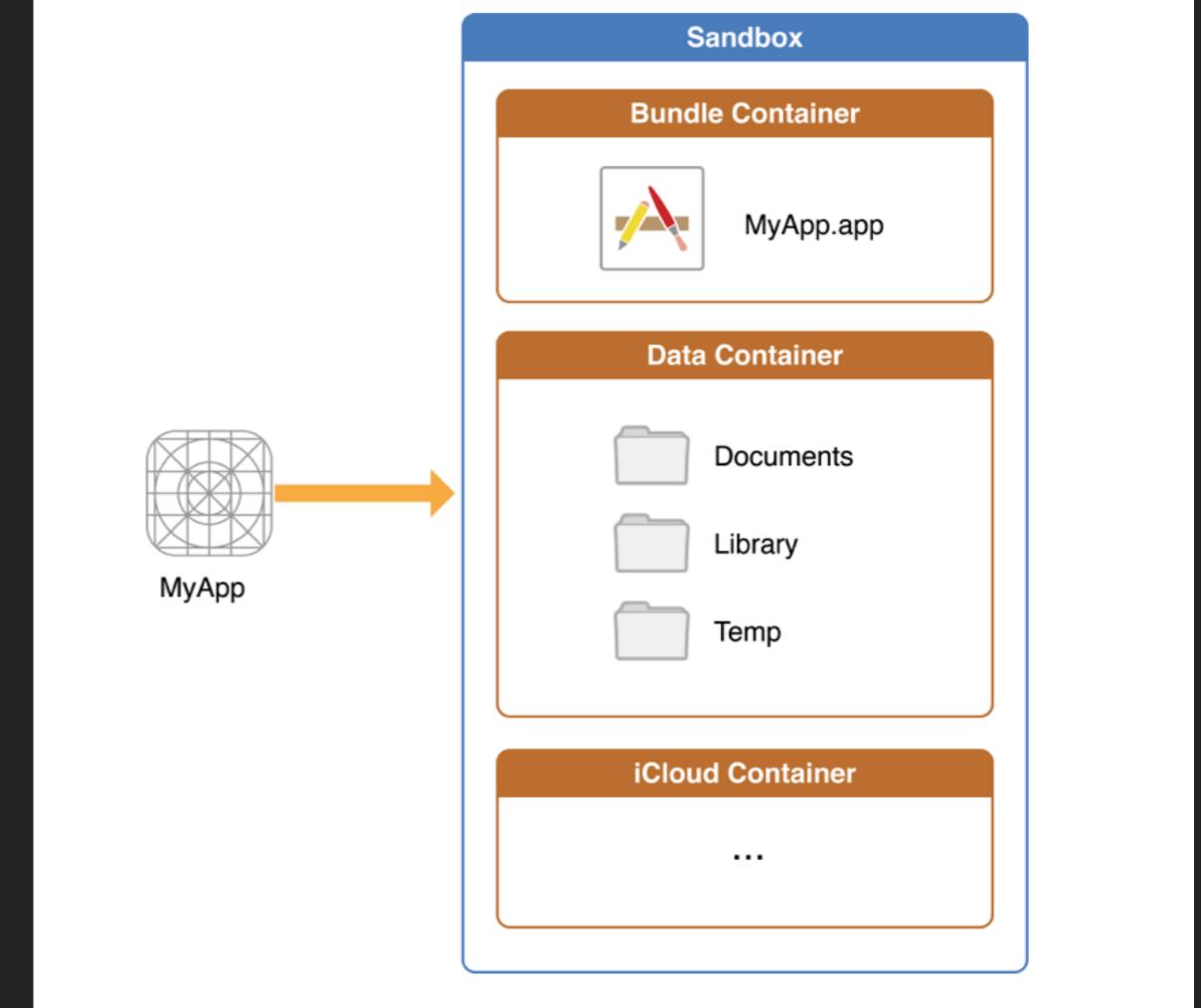
After the most recent episode is played, you'll return to older episodes.

Episodes will not be automatically downloaded but will be deleted when played based on your Podcasts Settings.

FILE MANAGER

- ▶ Every app can create and access files inside its sandbox directory
 - ▶ An app cannot access another app's files

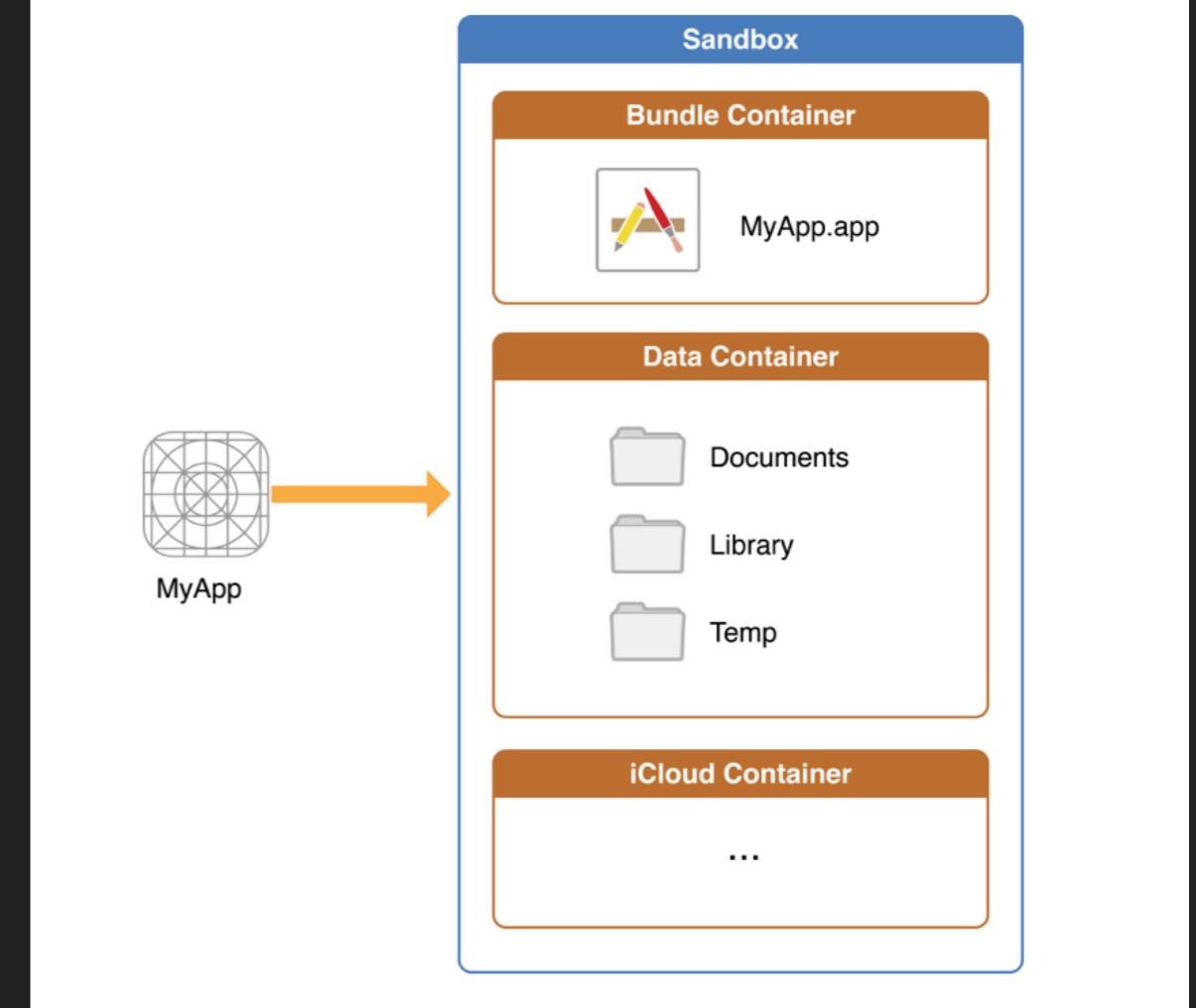
Figure 1–1 An iOS app operating within its own sandbox directory



FILE MANAGER

- ▶ Documents
 - ▶ Files containing user data
- ▶ Library
 - ▶ Files that don't contain user data
- ▶ Temp
 - ▶ Temporary files

Figure 1–1 An iOS app operating within its own sandbox directory

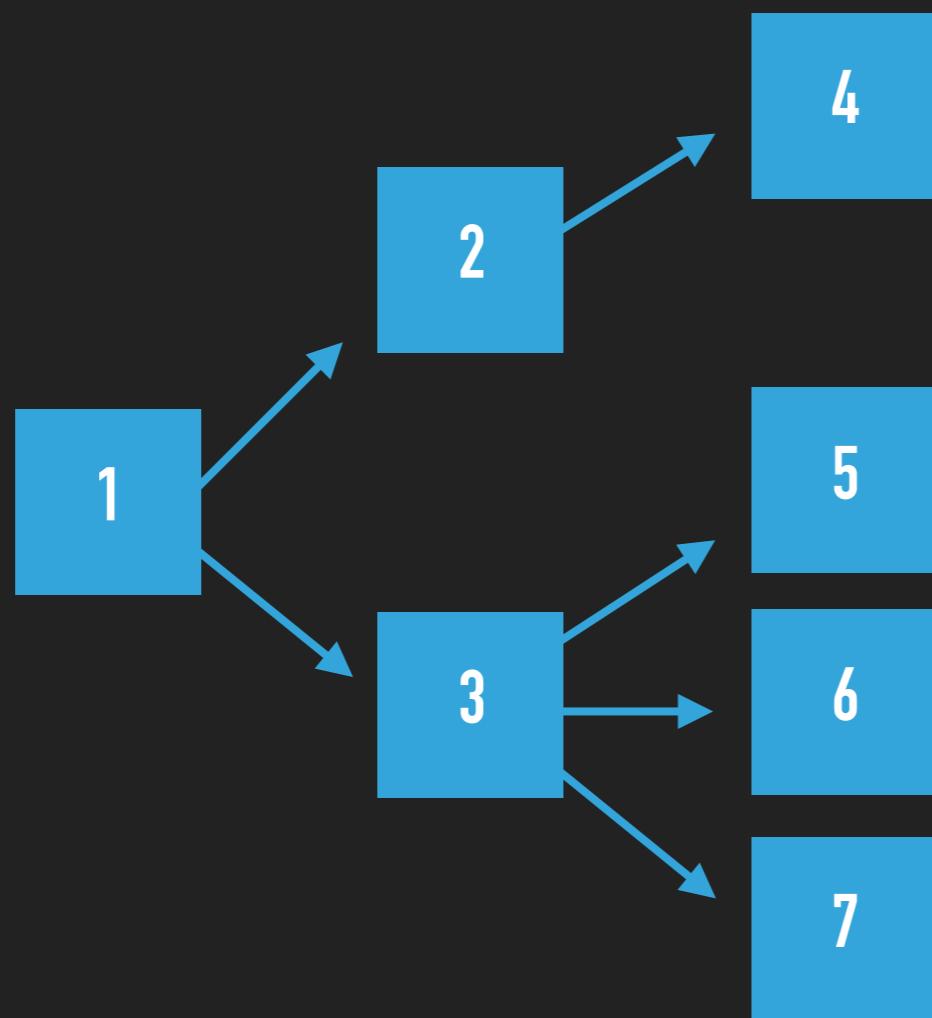


FILE MANAGER

- ▶ You can store any type of file
 - ▶ Images
 - ▶ Text
 - ▶ Audio
 - ▶ Archived data

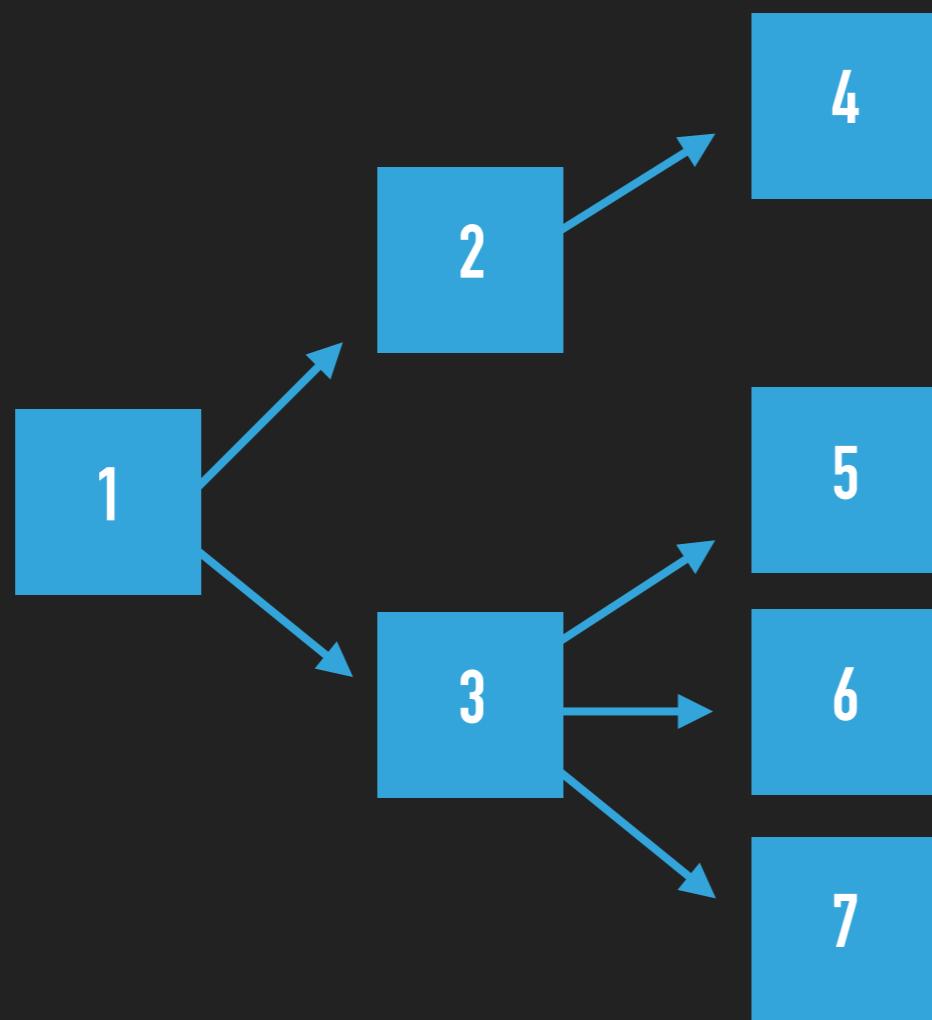
CORE DATA

- ▶ Useful if you have many objects related to each other in complex ways



CORE DATA

- ▶ Manages an object graph that you can traverse and query



CORE DATA

- ▶ Data persistence
 - ▶ Tracks which objects in memory have been created, deleted or changed
 - ▶ Performs automatic data migrations

CORE DATA

When it comes to modeling, querying, traversing and persisting complex object graphs, there is no substitute for Core Data. Core Data is a big hammer, but not every problem is a nail—much less a sufficiently large nail.

~ NSHipster (<https://nshipster.com/nsencoding/>)

iCLOUD

- ▶ Apple's cloud-based persistence option

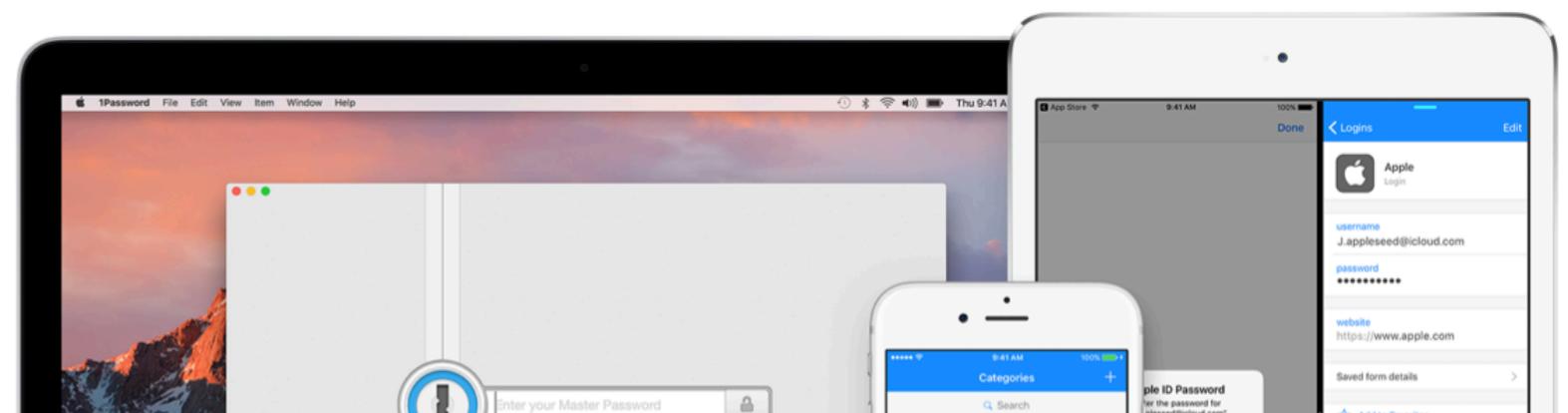
iCloud for Developers

Securely store your app's data and documents in iCloud — and keep them up to date across macOS, iOS, watchOS, tvOS, and the web. With iCloud, your users can access the information they want, wherever they want it.



CloudKit

CloudKit keeps your app data connected and up to date. And CloudKit JS makes it easy to sync data to the cloud.



iCLOUD

- ▶ Pros:
 - ▶ Reasonable amount of storage included with paid developer account
 - ▶ Automatically sync across a user's devices
 - ▶ Users don't have to create new accounts

iCLOUD

- ▶ Cons:
 - ▶ Data isn't accessible outside of Apple's ecosystem

MORE CONCEPTS IN

SWIFT

CONVENIENCE

INITIALIZERS

CONVENIENCE INITIALIZER

- ▶ A **convenience initializer** is one that calls another initializer on the same class
- ▶ This helps you avoid duplicate code
 - ▶ Easier to read
 - ▶ Less bug-prone

CONVENIENCE INITIALIZER

- ▶ An initializer that doesn't call any other initializers is a **designated initializer**
- ▶ A class can have multiple designated and convenience initializers

CONVENIENCE INITIALIZER

```
3 class Person {  
4  
5     let firstName: String  
6     let lastName: String  
7     let fullName: String  
8  
9     // designated initializer  
10    init(firstName: String, lastName: String, fullName: String) {  
11        self.firstName = firstName  
12        self.lastName = lastName  
13        self.fullName = fullName  
14    }  
15  
16    // convenience initializer  
17    convenience init(firstName: String, lastName: String) {  
18        let fullName = "\(firstName) \(lastName)"  
19        self.init(firstName: firstName, lastName: lastName, fullName: fullName)  
20    }  
21  
22}
```



CONVENIENCE INITIALIZER MUST CALL THE DESIGNATED INITIALIZER

FAILABLE

INITIALIZERS

FAILABLE INITIALIZERS

- ▶ Sometimes you want to allow an initializer to fail to create an instance of an object
 - ▶ Example: `Int` has failable initializer that takes a string

```
38  
39 let a = Int("400")      400  
40 let b = Int("banana")   nil  
▶  
42
```

FAILABLE INITIALIZERS

- ▶ Failable initializers always return an optional
 - ▶ If the initializer succeeds, the optional contains a value
 - ▶ Otherwise, the object is nil

38		
39	let a = Int("400")	400
40	let b = Int("banana")	nil
41	(play icon)	
42		

FAILABLE INITIALIZERS

```
20 class Cat {  
21  
22     let name: String  
23     let age: Int  
24  
25     // failable initializer  
26     init?(name: String, age: Int) {  
27         guard !name.isEmpty, age > 0 else { return nil }  
28         self.name = name  
29         self.age = age  
30     }  
31 }  
32 }
```



INIT? INDICATES THAT THE INITIALIZER IS FAILABLE

ENUMS WITH

RAW VALUES

ENUMS WITH RAW VALUES

- ▶ Each case of an enum can be associated with a raw value

DECLARING THE RAW
VALUE'S TYPE

```
3 enum State: String {  
4     case illinois = "IL"  
5     case indiana = "IN"  
6     case michigan = "MI"  
7     case minnesota = "MN"  
8     case ohio = "OH"  
9     case wisconsin = "WI"  
10 }  
11
```

ENUMS WITH RAW VALUES

- ▶ Raw values can be strings, characters or numbers
- ▶ Each case needs to have the same type of raw value
(meaning, you can't mix and match `String` and `Int`)
- ▶ Each case needs a unique raw value (no duplicates)

ENUMS WITH RAW VALUES

- ▶ An enum with raw values automatically gets an initializer that receives a raw value as input
 - ▶ This initializer is failable!

```
14 let landOfLincoln = State(rawValue: "IL") illinois
15 let sunshineState = State(rawValue: "FL") nil
```



HANDLING ERRORS

DO-TRY-CATCH

REPRESENTING ERRORS

- ▶ Errors are types that conform to the `Error` protocol
- ▶ Enums work well for representing related error conditions

```
8  enum CheckoutError: Error {  
9      case titleNotFound  
10     case titleUnavailable  
11 }
```

WHY THROW AN ERROR?

- ▶ Why throw an error when you could just return `nil`?

THROWING ERRORS

- ▶ A function that can throw an error instead of returning normally must be marked with **throws**

```
19     func checkout(title: String) throws -> Book {  
20         var requestedBook: Book?  
21         for book in books {  
22             if book.title == title {  
23                 requestedBook = book  
24             }  
25         }  
26  
27         guard let book = requestedBook else {  
28             throw CheckoutError.titleNotFound  
29         }  
30  
31         guard book.isAvailable else {  
32             throw CheckoutError.titleUnavailable  
33         }  
34  
35         return book  
36     }  
37
```



HANDLING ERRORS

- ▶ Use do-try-catch when calling a function that throws
 - ▶ do block must contain a function that throws
 - ▶ Mark throwing function with try
 - ▶ catch block only executes if an error is actually thrown

```
41 do {  
42     let book1 = try library.checkout(title: "The DaVinci Code")  
43     print(book1.title)  
44 } catch (let error) {  
45     print(error)  
46 }
```

HANDLING ERRORS

- ▶ You can also choose to ignore potential errors
 - ▶ `try?` means return nil if an error is thrown
 - ▶ `try!` means crash if an error is thrown

```
48 let book2 = try? library.checkout(title: "Where the Wild Things Are")
49 let book3 = try! library.checkout(title: "The Hobbit")
50
```

SAVING BASIC TYPES WITH

USER DEFAULTS

SAVING A VALUE

- ▶ Associate a value with a key when saving

```
UserDefaults.standard.set(nameTextField.text, forKey: "name")
```

RETRIEVING A VALUE

- ▶ Retrieve the value using the same key

```
nameTextField.text = UserDefaults.standard.string(forKey: "name")
```

REMOVING A VALUE

- ▶ Remove the value for a key

```
UserDefaults.standard.removeObject(forKey: "name")
```

TYPES THAT CAN BE SAVED

- ▶ `NSData`, `NSString`, `NSNumber`, `NSDate`, `NSArray`, `NSDictionary`
- ▶ All of these types have toll-free bridging to their Swift counterparts

SETTING DEFAULT VALUES

Setting Default Values

func `set`(Any?, forKey: String)

Sets the value of the specified default key.

func `set`(Float, forKey: String)

Sets the value of the specified default key to the specified float value.

func `set`(Double, forKey: String)

Sets the value of the specified default key to the double value.

func `set`(Int, forKey: String)

Sets the value of the specified default key to the specified integer value.

func `set`(Bool, forKey: String)

Sets the value of the specified default key to the specified Boolean value.

func `set`(URL?, forKey: String)

Sets the value of the specified default key to the specified URL.

GETTING DEFAULT VALUES

Getting Default Values

func `object(forKey: String) -> Any?`

Returns the object associated with the specified key.

func `url(forKey: String) -> URL?`

Returns the URL associated with the specified key.

func `array(forKey: String) -> [Any]?`

Returns the array associated with the specified key.

func `dictionary(forKey: String) -> [String : Any]?`

Returns the dictionary object associated with the specified key.

func `string(forKey: String) -> String?`

Returns the string associated with the specified key.

func `stringArray(forKey: String) -> [String]?`

Returns the array of strings associated with the specified key.

func `data(forKey: String) -> Data?`

Returns the data object associated with the specified key.

func `bool(forKey: String) -> Bool`

REMOVING DEFAULT VALUES

Removing Defaults

func `removeObject(forKey: String)`

Removes the value of the specified default key.

SAVING CUSTOM TYPES WITH

USER DEFAULTS

SAVING CUSTOM TYPES

- ▶ What if I want to save an object that is not one of the supported types?
 - ▶ Supported types: **NSData**, **NSString**, **NSNumber**, **NSDate**, **NSArray**, **NSDictionary**

SAVING CUSTOM TYPES

- ▶ To save an instance of a custom type, convert it into `NSData`
- ▶ Conforming to the `NSCoding` protocol makes it easy to convert between your custom type and `NSData`

NSCODING

- ▶ NSCoding has two methods that you must implement

```
protocol NSCoding {  
    init?(coder aDecoder: NSCoder)  
    func encode(with aCoder: NSCoder)  
}
```

NSOBJECT

- ▶ You also need to subclass `NSObject` , which allows the object to be treated like an Objective C type

IMPLEMENTING NSCODING

```
class Book: NSObject, NSCoding {

    let title: String
    let author: String
    let category: BookCategory

    init?(title: String?, author: String?, category: BookCategory?) { ... }

    required convenience init?(coder aDecoder: NSCoder) {
        let title = aDecoder.decodeObject(forKey: "title") as? String
        let author = aDecoder.decodeObject(forKey: "author") as? String
        let categoryRawValue = aDecoder.decodeInteger(forKey: "category")
        let category = BookCategory(rawValue: categoryRawValue)

        self.init(title: title, author: author, category: category)
    }

    func encode(with aCoder: NSCoder) {
        aCoder.encode(title, forKey: "title")
        aCoder.encode(author, forKey: "author")
        aCoder.encode(category.rawValue, forKey: "category")
    }
}
```

NSKEYEDARCHIVER

- ▶ Use `NSKeyedArchiver` to convert the object into `Data`

```
do {
    let data = try NSKeyedArchiver.archivedData(withRootObject: books,
                                                requiringSecureCoding: false)

    UserDefaults.standard.set(data, forKey: "books")
} catch (let error) {
    print("Error saving books to user defaults: \(error)")
}
```

NSKEYEDUNARCHIVER

- ▶ Use `NSKeyedUnarchiver` to unarchive the `Data` back into your custom object

```
guard let data = UserDefaults.standard.data(forKey: "books") else { return [] }

var books: [Book]?
do {
    books = try NSKeyedUnarchiver.unarchiveTopLevelObjectWithData(data) as? [Book]
} catch (let error) {
    print("Error fetching books from user defaults: \(error)")
}

return books ?? []
```

SAVING TO DISK WITH

FILE MANAGER

FILE MANAGER

- ▶ “A convenient interface to the contents of the file system, and the primary means of interacting with it.”
~ Apple Documentation

FILE MANAGER

- ▶ Makes it easy to get the path to your app's Documents Directory, where you can store data to files

```
guard let documentDirectory = FileManager.default.urls(for: .documentDirectory,  
                                                    in: .userDomainMask).first else { return }  
  
let url = documentDirectory.appendingPathComponent("books")
```

FILE MANAGER

- ▶ Convert your custom objects to data with `NSKeyedArchiver`
- ▶ Write data to your new url in the Documents Directory

```
do {  
    let data = try NSKeyedArchiver.archivedData(withRootObject: books, requiringSecureCoding: true)  
    try data.write(to: url)  
} catch (let error) {  
    print("Error saving books to file: \(error)")  
}
```

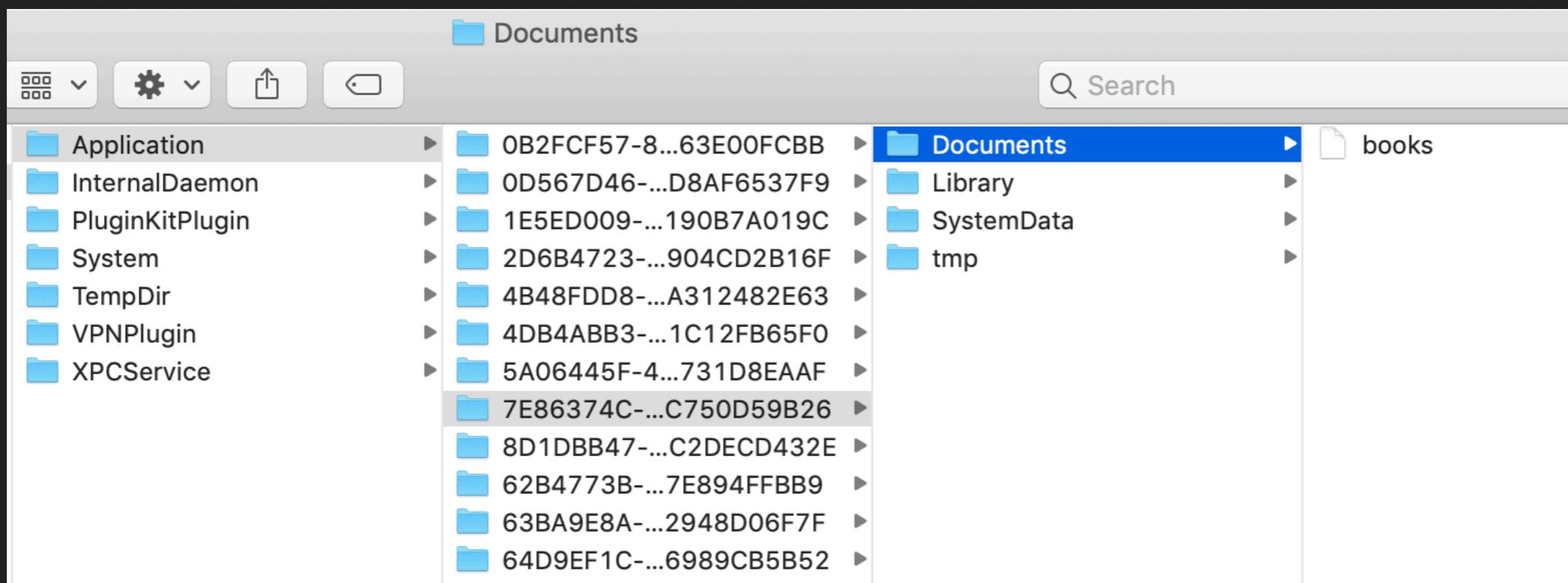
FILE MANAGER

- ▶ Fetch data from the file
- ▶ Convert the data back into your custom objects with **NSKeyedUnarchiver**

```
var books: [Book]?
do {
    let data = try Data(contentsOf: url)
    books = try NSKeyedUnarchiver.unarchiveTopLevelObjectWithData(data) as? [Book]
} catch (let error) {
    print("Error fetching books from file: \(error)")
}
```

TIP #1

- ▶ To navigate to your simulator's Document Directory:
 - ▶ Print the path to the Xcode console and copy it
 - ▶ In Finder, press **Shift+Command+G** and paste the path



TIP #2

- ▶ Deleting the app will delete its data from the Documents Directory and User Defaults
 - ▶ You'll need to do this if you change the structure of the data you're saving
 - ▶ You can also manually delete a file from the Documents Directory if you're using a simulator

TIP #3

- ▶ Every time you run your app on a different simulator, you get a new sandbox
 - ▶ New Documents Directory
 - ▶ New User Defaults
- ▶ In other words, data is not shared between simulators

TIP #4

- ▶ Make sure to test that your app works the **first time** the user runs it
 - ▶ There won't be anything in the Documents Directory or User Defaults

BUILDING FORMS FOR

USER INPUT

CONTROLS FOR USER INPUT

- ▶ iOS provides many controls that you can use to allow users to input data
- ▶ Examples:
 - ▶ Segmented control
 - ▶ Slider
 - ▶ Switch
 - ▶ Stepper
 - ▶ Date Picker

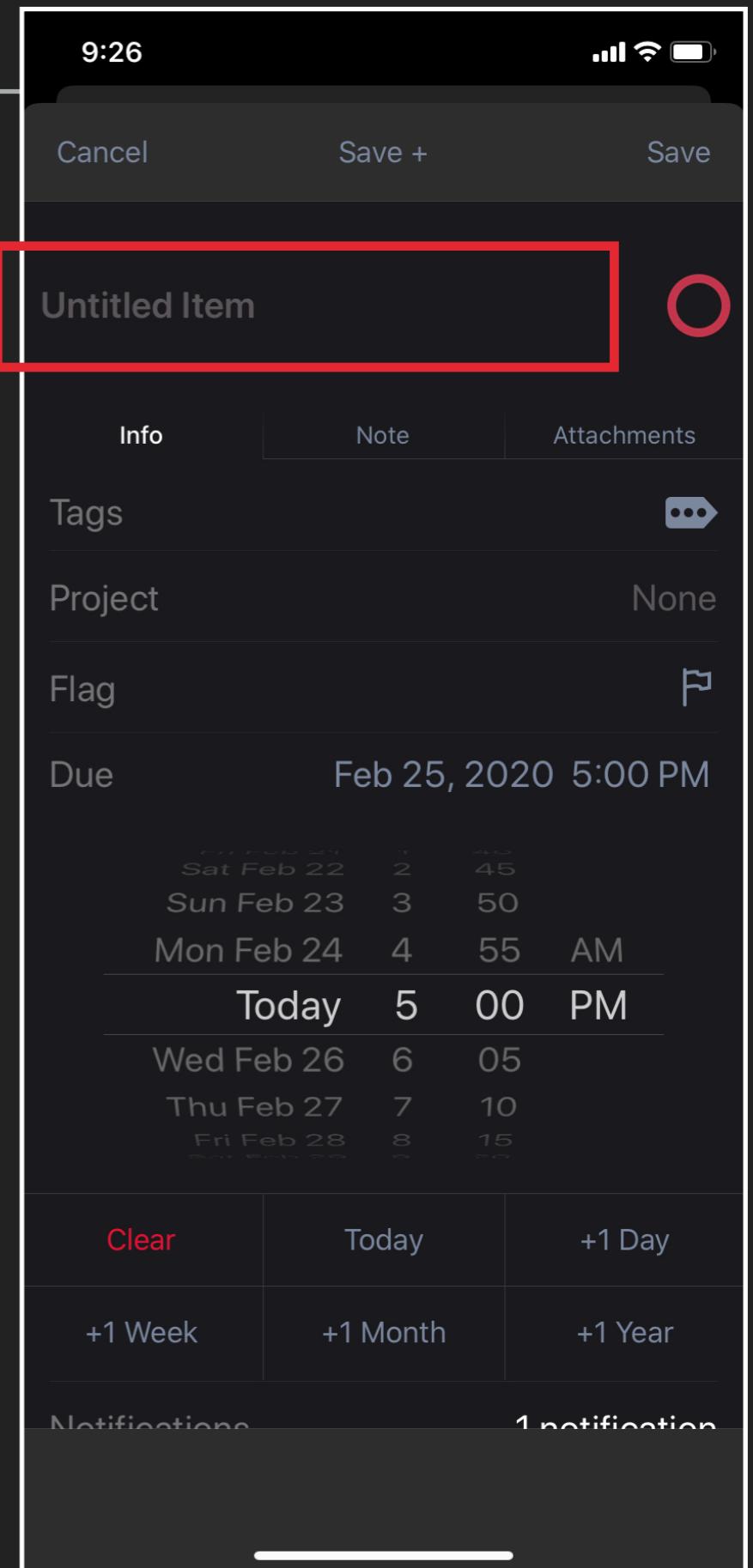
CONTROLS FOR USER INPUT

- ▶ There are three main controls that involve text input:
 - ▶ UITextField
 - ▶ UITextView
 - ▶ UISearchBar

USER INPUT

TEXT FIELD

- ▶ UITextField displays a single line of editable text

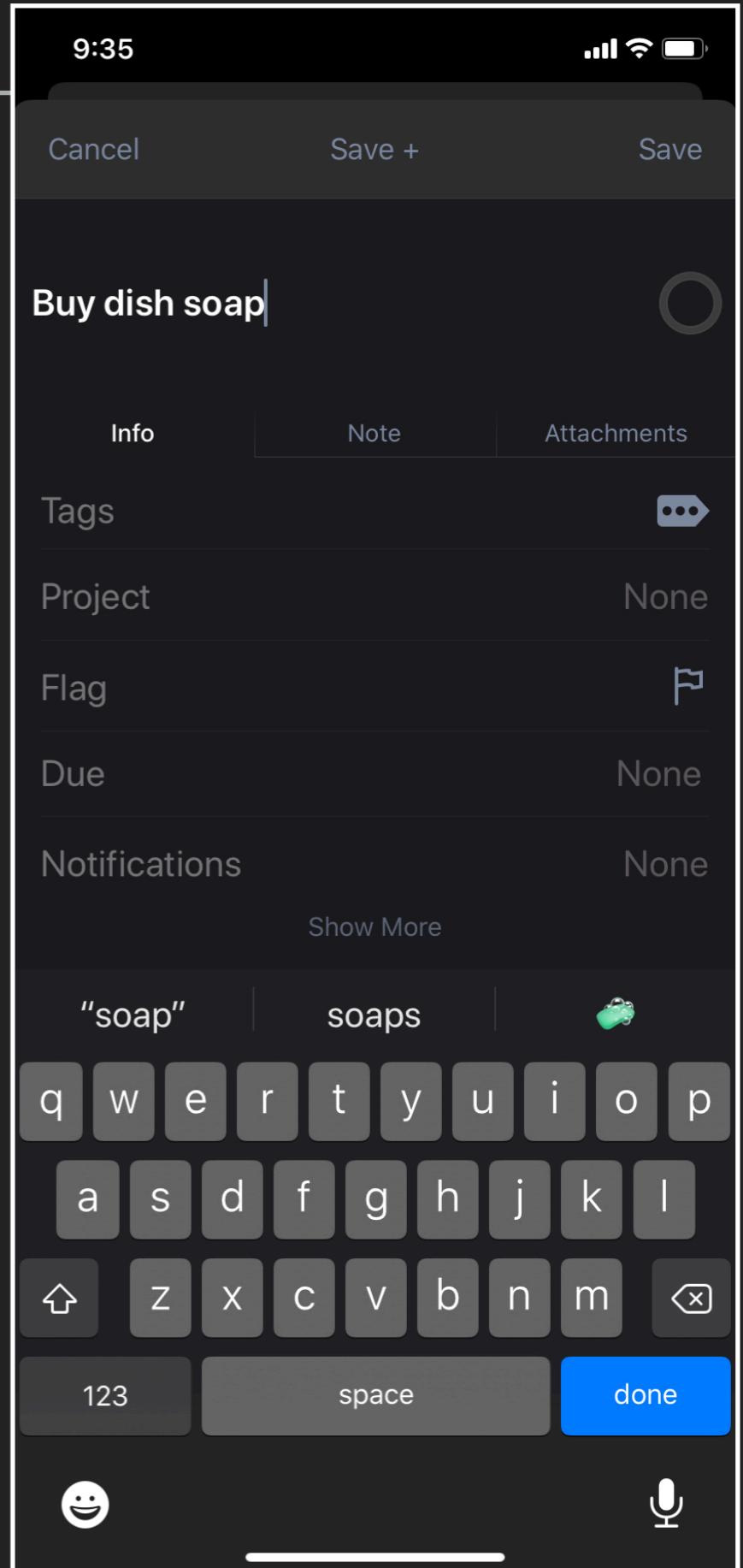


OmniFocus

USER INPUT

TEXT FIELD

- ▶ When the user taps inside of a text field, the keyboard automatically appears
- ▶ The text field becomes the **first responder**



OmniFocus

TEXT FIELD

- ▶ You can programmatically show and hide the keyboard by calling these methods:

```
textField.becomeFirstResponder()
```

```
textField.resignFirstResponder()
```

TEXT FIELD

- ▶ Even though the keyboard *shows* automatically, it does not *hide* automatically
- ▶ You may want to programmatically hide the keyboard when:
 - ▶ The user taps “Return”
 - ▶ The user taps away from the text field
 - ▶ The user drags down on the keyboard

TEXT FIELD

- ▶ Use UITextFieldDelegate to be notified of important events related to the text field
- ▶ For example:
 - ▶ Editing began or ended
 - ▶ Characters were added or deleted
 - ▶ User tapped the “Return” or “Clear” buttons

TEXT FIELD

- ▶ Use `textFieldShouldReturn` to dismiss the keyboard when the user taps on the “Return” button

```
32  
33 extension AddBookViewController: UITextFieldDelegate {  
34  
35     func textFieldShouldReturn(_ textField: UITextField) -> Bool {  
36         textField.resignFirstResponder()  
37     }  
38 }  
39
```

TEXT FIELD

- ▶ Use a tap gesture recognizer to hide the keyboard when the user taps away from the text field
- ▶ Call `endEditing` on the view to have any text field in its view hierarchy resign first responder

```
@objc func handleTap(_ sender: UITapGestureRecognizer) {  
    // Dismiss the keyboard when the user taps away from a text field  
    tableView.endEditing(true)  
}
```

TEXT FIELD

- ▶ If the text field is inside of a scroll view, you can set the scroll view's `keyboardDismissMode`

`case none`

The keyboard does not get dismissed with a drag.

`case onDrag`

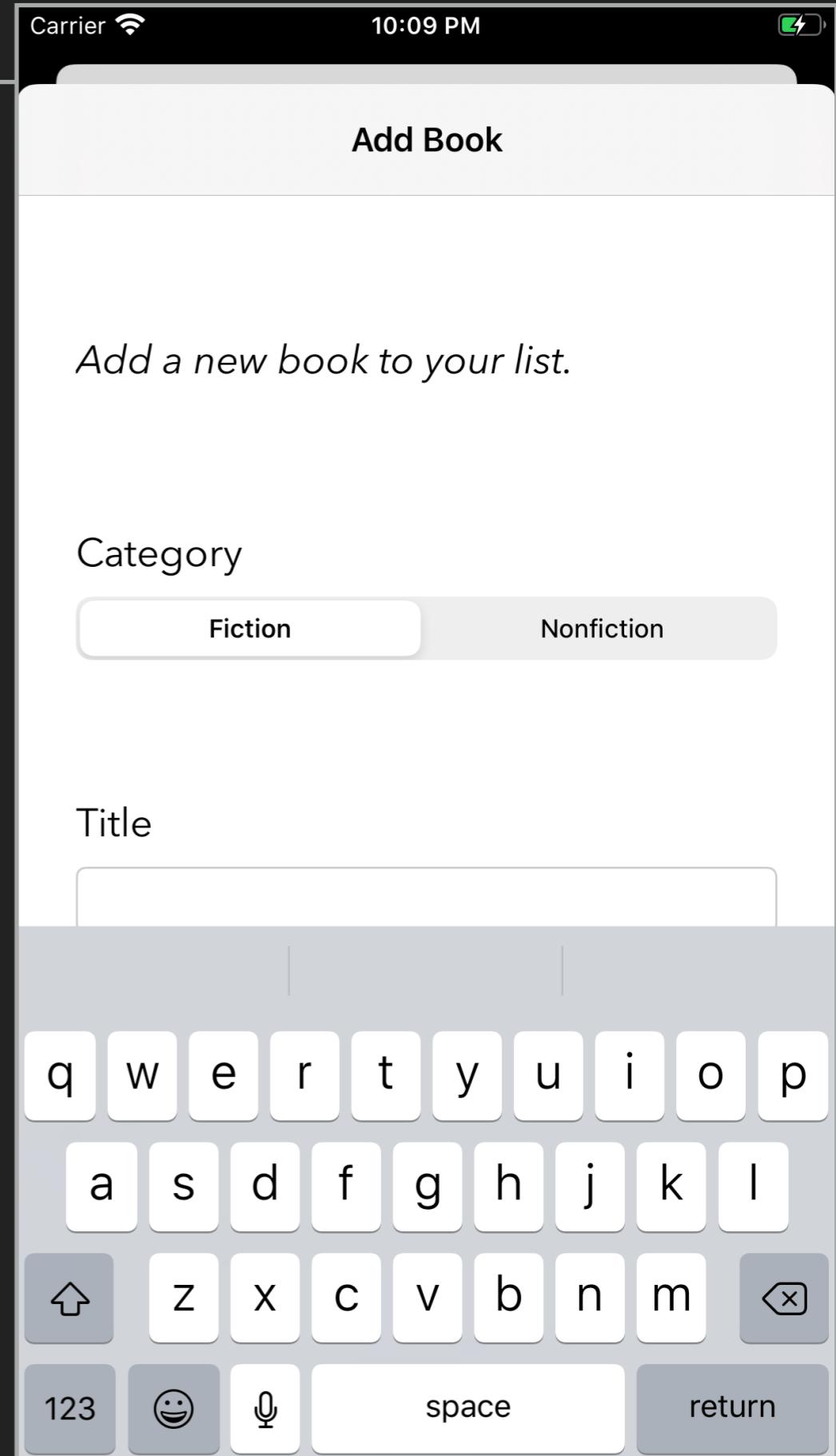
The keyboard is dismissed when a drag begins.

`case interactive`

The keyboard follows the dragging touch offscreen, and can be pulled upward again to cancel the dismiss.

TEXT FIELD

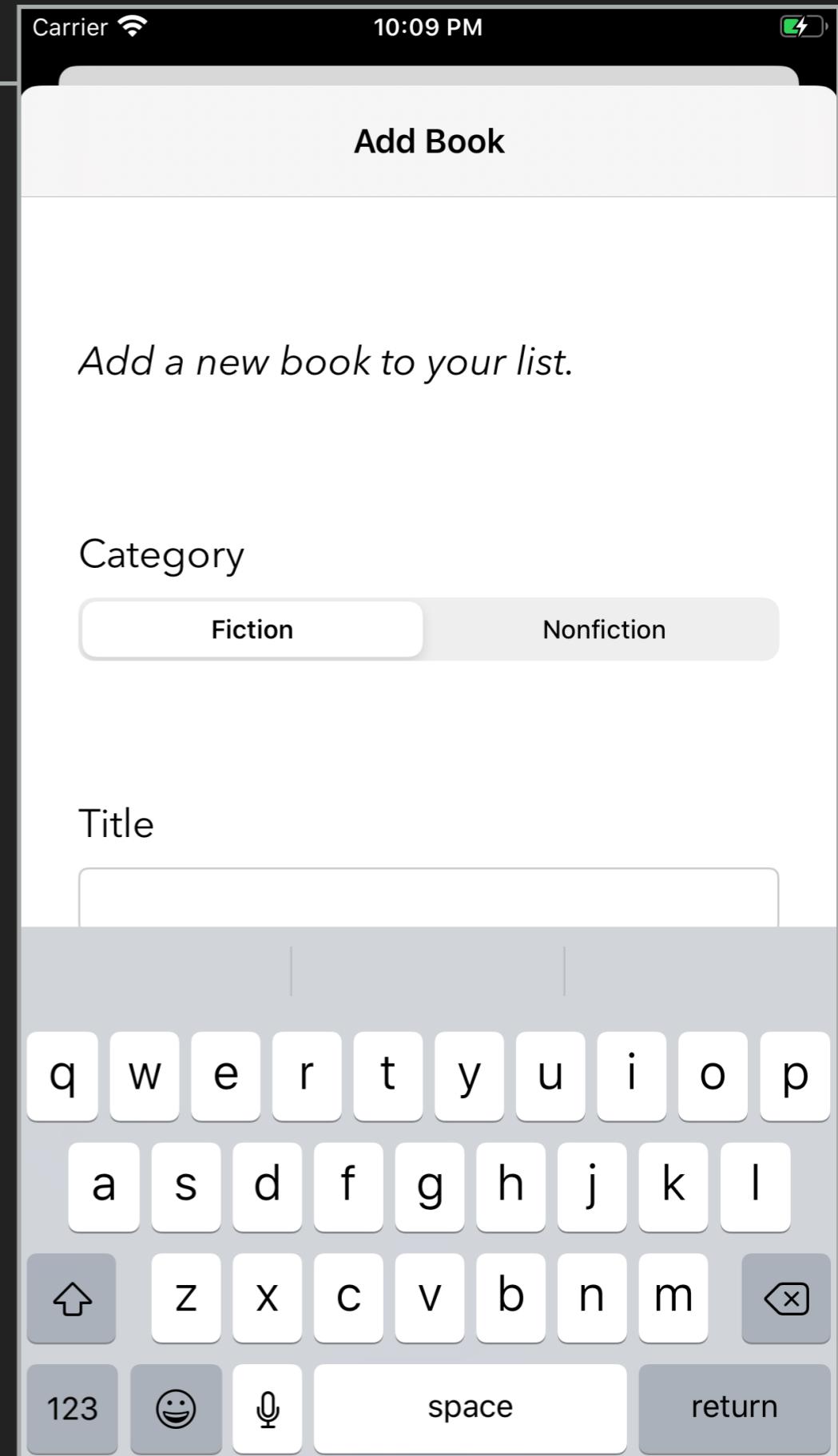
- ▶ What do we do if the keyboard covers important parts of the user interface? 😢



USER INPUT

TEXT FIELD

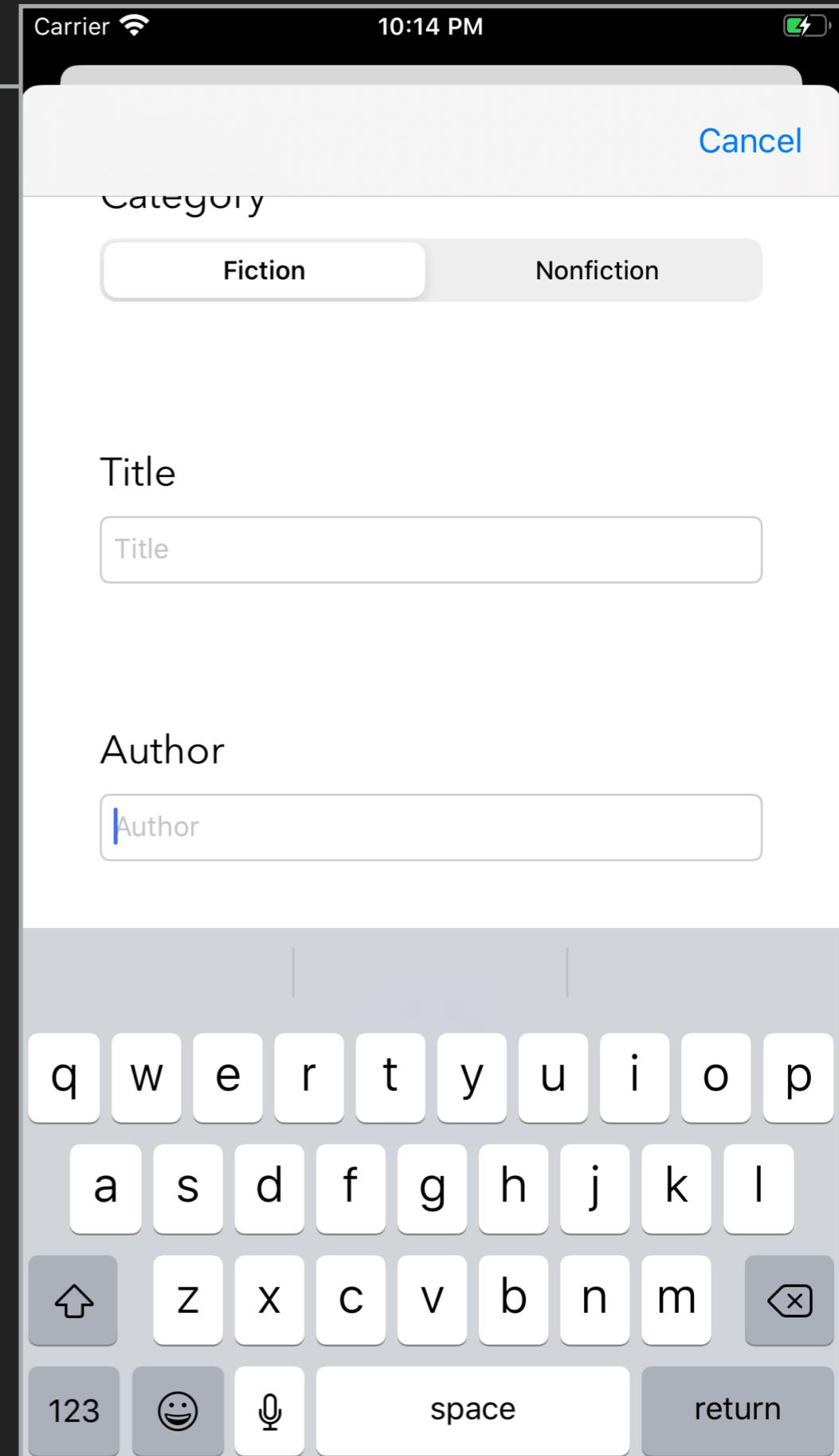
- ▶ Option 1: Embed everything in a UIScrollView and adjust the scroll view when the keyboard appears
- ▶ Note - this can be pretty tricky!



USER INPUT

TEXT FIELD

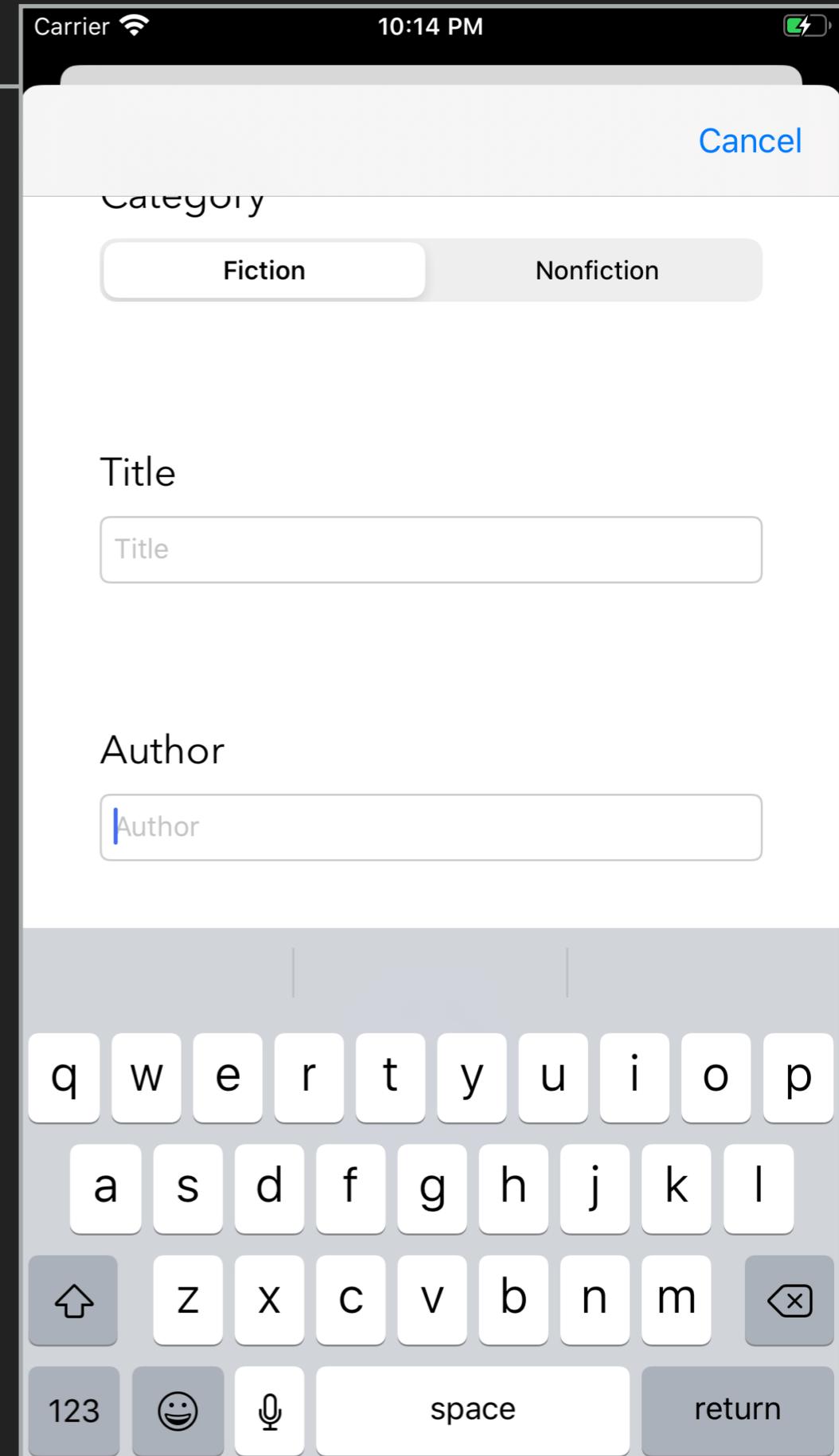
- ▶ Option 2: Use a UITableViewController
- ▶ Text fields will scroll automatically! 



USER INPUT

TEXT FIELD

- ▶ Option 2: Use a UITableViewController
- ▶ Text fields will scroll automatically! 

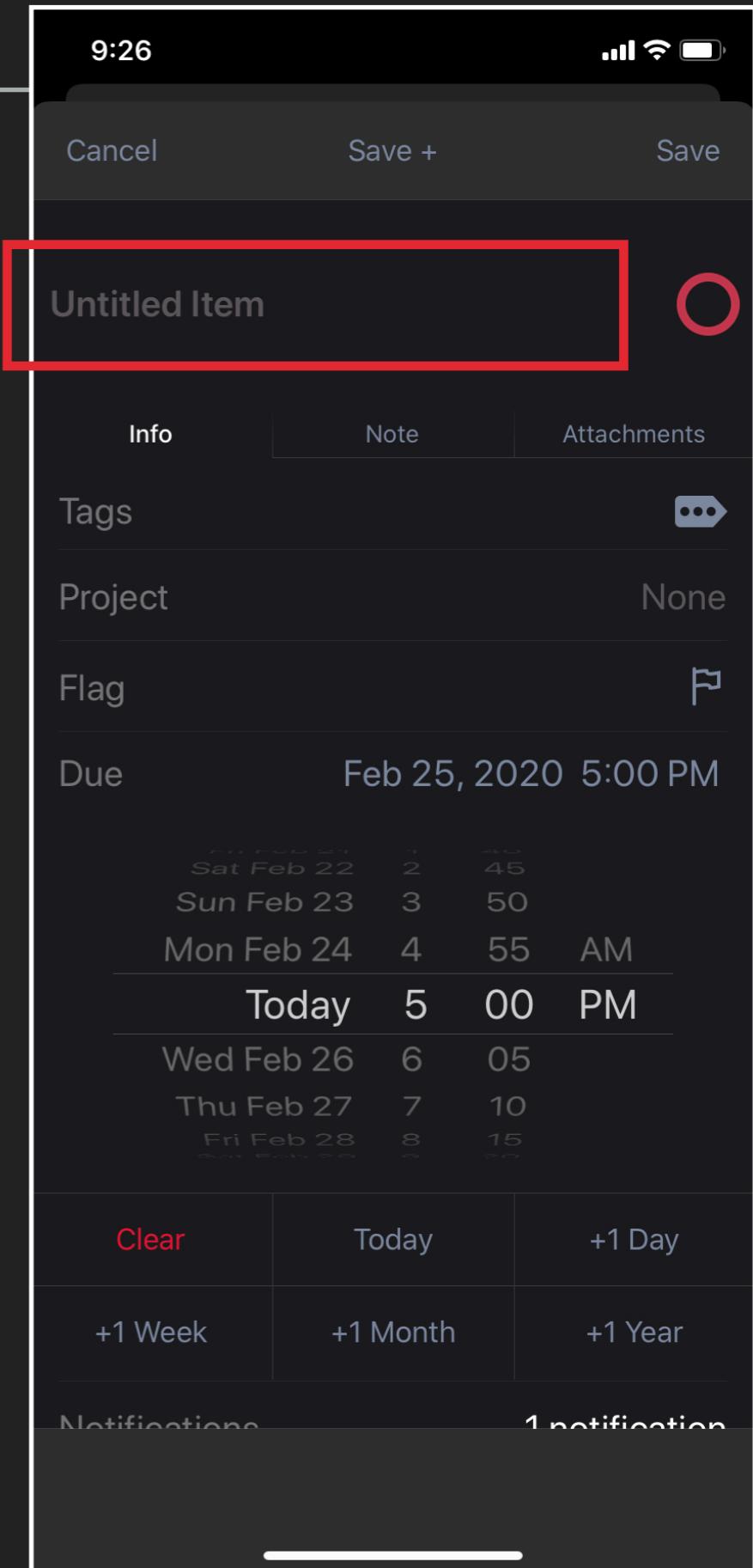


TEXT FIELD

- ▶ Since most people don't *love* typing on mobile devices, here are a few more things you can do to make the experience as smooth as possible...

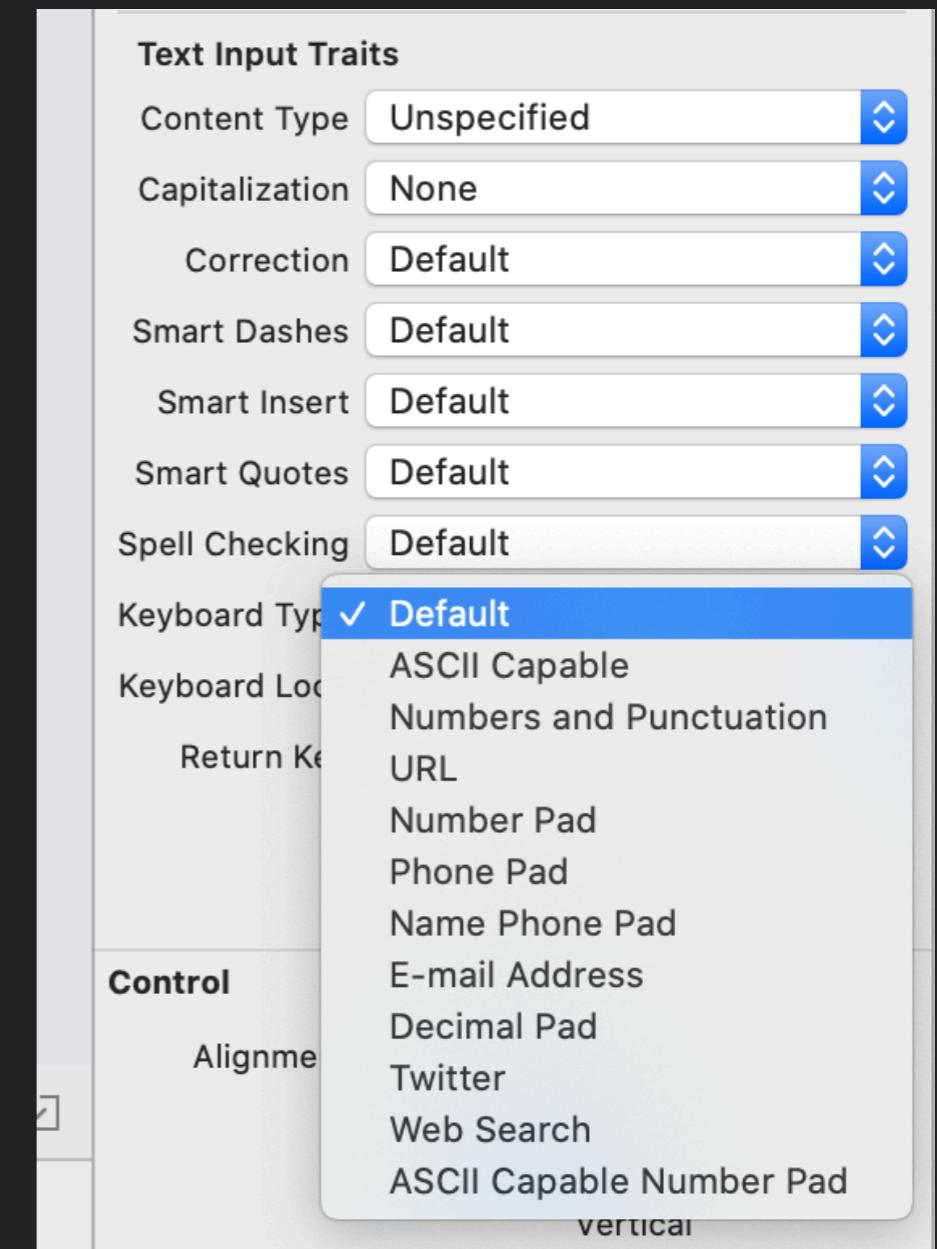
TEXT FIELD

- ▶ Display **placeholder text** in the text field
 - ▶ Different font color
 - ▶ Disappears when the user starts typing
 - ▶ Provides context about the purpose of the field



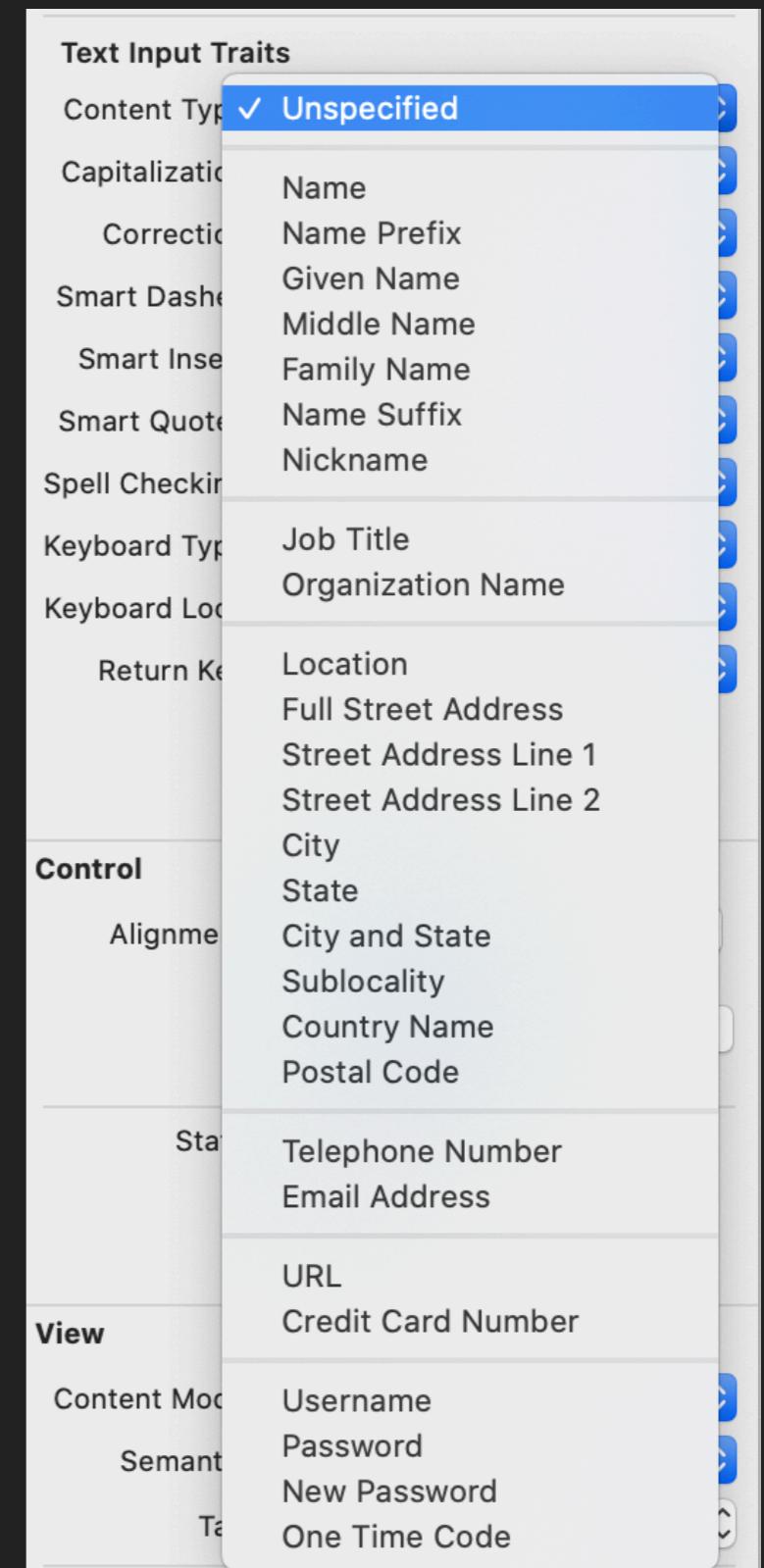
TEXT FIELD

- ▶ Use the correct keyboard type
- ▶ (Don't show letters when the field only allows numbers)



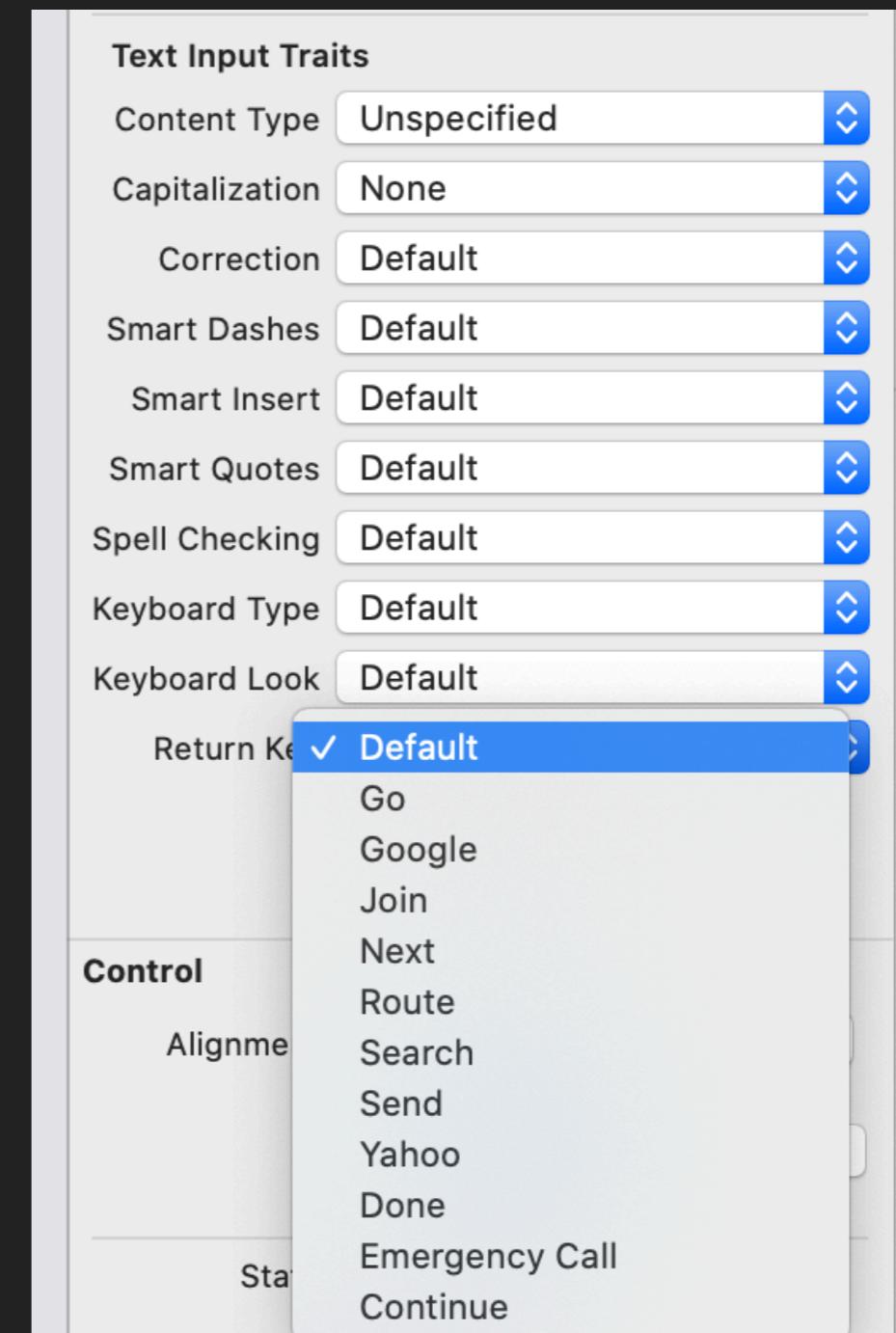
TEXT FIELD

- ▶ Set the **content type** whenever possible
- ▶ The keyboard will show suggestions, allowing users to quickly fill in basic info



TEXT FIELD

- ▶ Show the appropriate text on the return key
- ▶ If you have multiple text fields, automatically move the user to the next one



TEXT FIELD

- ▶ Most importantly, don't allow your users to get "stuck"
 - ▶ Make it easy to dismiss the keyboard
 - ▶ Scroll text fields up so that they're above the keyboard

CREATING A

SETTINGS BUNDLE

SETTINGS BUNDLE

SETTINGS APP

- ▶ Your app can have its own section in the Settings App

Slack

7:17



Settings

Slack

ALLOW SLACK TO ACCESS

Camera



Siri & Search
Siri & Suggestions



Background App Refresh



Cellular Data



SLACK SETTINGS

TROUBLESHOOTING

Reset cache on next launch

Send Feedback for Sign-in

SETTINGS BUNDLE

SETTINGS APP

- ▶ This is where the user can change your app's access to things like:
 - ▶ Location
 - ▶ Camera
 - ▶ Contacts
 - ▶ Notifications

Google Maps

7:18



Settings

Google Maps

ALLOW GOOGLE MAPS TO ACCESS

- Location While Using >
- Siri & Search Siri & Suggestions >
- Notifications Banners, Sounds, Badges >
- Background App Refresh
- Cellular Data

SETTINGS BUNDLE

SETTINGS APP

- ▶ You can also include settings specific to your app
 - ▶ You do this by creating a **Settings Bundle**

Token

7:19



Settings

Token

ALLOW TOKEN TO ACCESS



Camera



Siri & Search

Siri & Suggestions



Cellular Data



TOKEN SETTINGS

RSA SECURID SOFTWARE TOKEN

Version

2.4.7

Mask PIN



SETTINGS BUNDLE

SETTINGS APP

- ▶ Of course, you could also choose to place custom settings in the app itself

Slack

7:21



Settings

Edit Profile

Susan Stevens >

Your Availability

Active

Notifications

>

Do Not Disturb

On >

Accessibility >

>

Language >

>

Advanced >

>

Apps & Integrations >

>

Analytics >

>

Send Feedback >

>

Rate Slack >

>

Help Center >

>

About

19.2.2 >

Join Beta >

>

SETTINGS APP

- ▶ Pros of using a Settings Bundle
 - ▶ You don't have to build a whole new screen
 - ▶ It eliminates potential clutter from your app

SETTINGS APP

- ▶ Cons of using a Settings Bundle
 - ▶ Users may not know that it's there
 - ▶ You get a limited number of controls

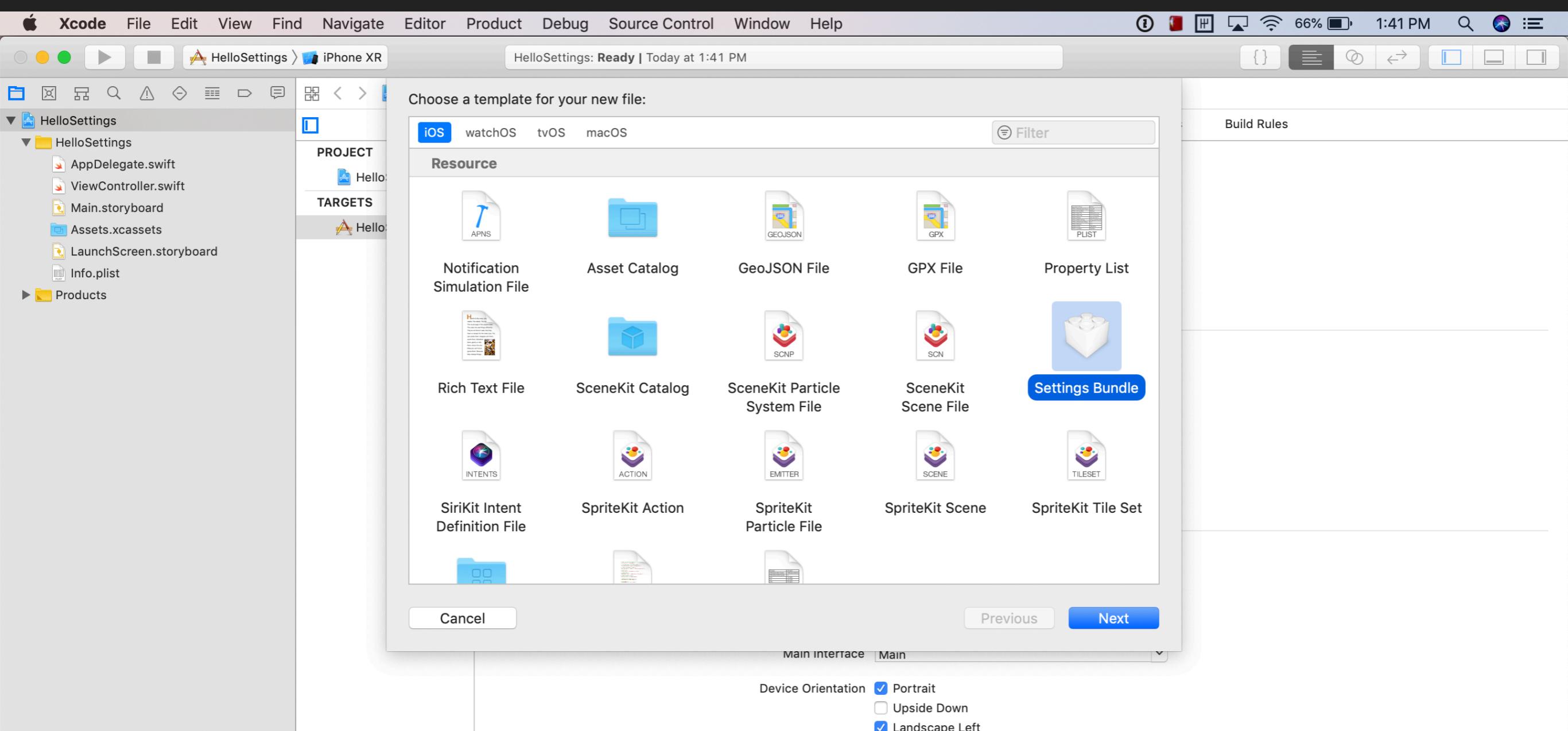
SETTINGS APP

- ▶ Secret use-case for a Settings Bundle: developing and testing your app
 - ▶ Point to different backend environments
 - ▶ Turn on and off caching
 - ▶ Show and hide features

SETTINGS BUNDLE

TO ADD A SETTINGS BUNDLE

► File -> New -> File



SETTINGS BUNDLE

EDITING YOUR SETTINGS BUNDLE

- ▶ Expand Settings.bundle and select Root.plist
- ▶ plist = Property List

The screenshot shows the Xcode interface with the project "HelloSettings" selected. The "Root.plist" file is open in the editor. The plist structure is as follows:

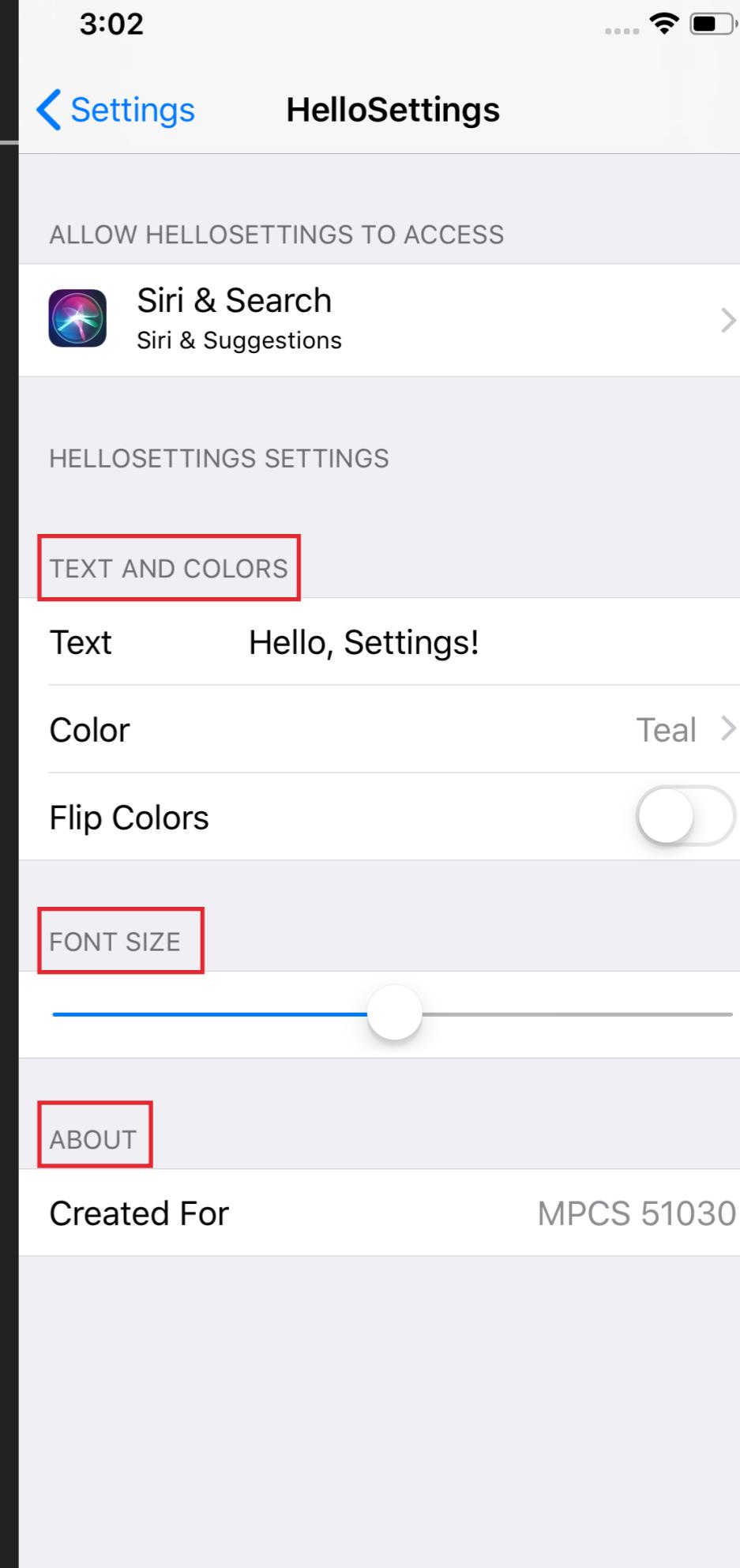
Key	Type	Value
iPhone Settings Schema	Dictionary	(2 items)
Strings Filename	String	Root
Preference Items	Array	(8 items)
Item 0 (Group - Text and Colors)	Dictionary	(2 items)
Type	String	Group
Title	String	Text and Colors
Item 1 (Text Field - Text)	Dictionary	(8 items)
Type	String	Text Field
Title	String	Text
Identifier	String	text
Default Value	String	Hello, Settings!
Text Field Is Secure	Boolean	NO
Keyboard Type	String	Alphabet
Autocapitalization Style	String	None
Autocorrection Style	String	No Autocorrection
Item 2 (Multi Value - Color)	Dictionary	(6 items)
Type	String	Multi Value
Title	String	Color
Identifier	String	colorId
Default Value	String	teal
Values	Array	(3 items)
Item 0	String	teal
Item 1	String	purple
Item 2	String	orange

SETTINGS BUNDLE

EDITING YOUR SETTINGS BUNDLE

- ▶ Under Preference Items, you can add:

- ▶ Groups



SETTINGS BUNDLE

EDITING YOUR SETTINGS BUNDLE

- ▶ Under Preference Items, you can add:
 - ▶ Titles (not editable by the user)

The screenshot shows the 'HelloSettings' settings screen. At the top, it says 'ALLOW HELLOSETTINGS TO ACCESS' with a 'Siri & Search' item and a 'Siri & Suggestions' button. Below that is a section titled 'HELLOSETTINGS SETTINGS' with a 'Text' field containing 'Hello, Settings!' and a 'Color' field set to 'Teal'. There's also a 'Flip Colors' switch. Further down is a 'FONT SIZE' slider. At the bottom, there's an 'ABOUT' section with 'Created For' and 'MPCS 51030' fields, both of which are highlighted with a red border.

3:02

Settings HelloSettings

ALLOW HELLOSETTINGS TO ACCESS

Siri & Search
Siri & Suggestions

HELLOSETTINGS SETTINGS

TEXT AND COLORS

Text Hello, Settings!

Color Teal >

Flip Colors

FONT SIZE

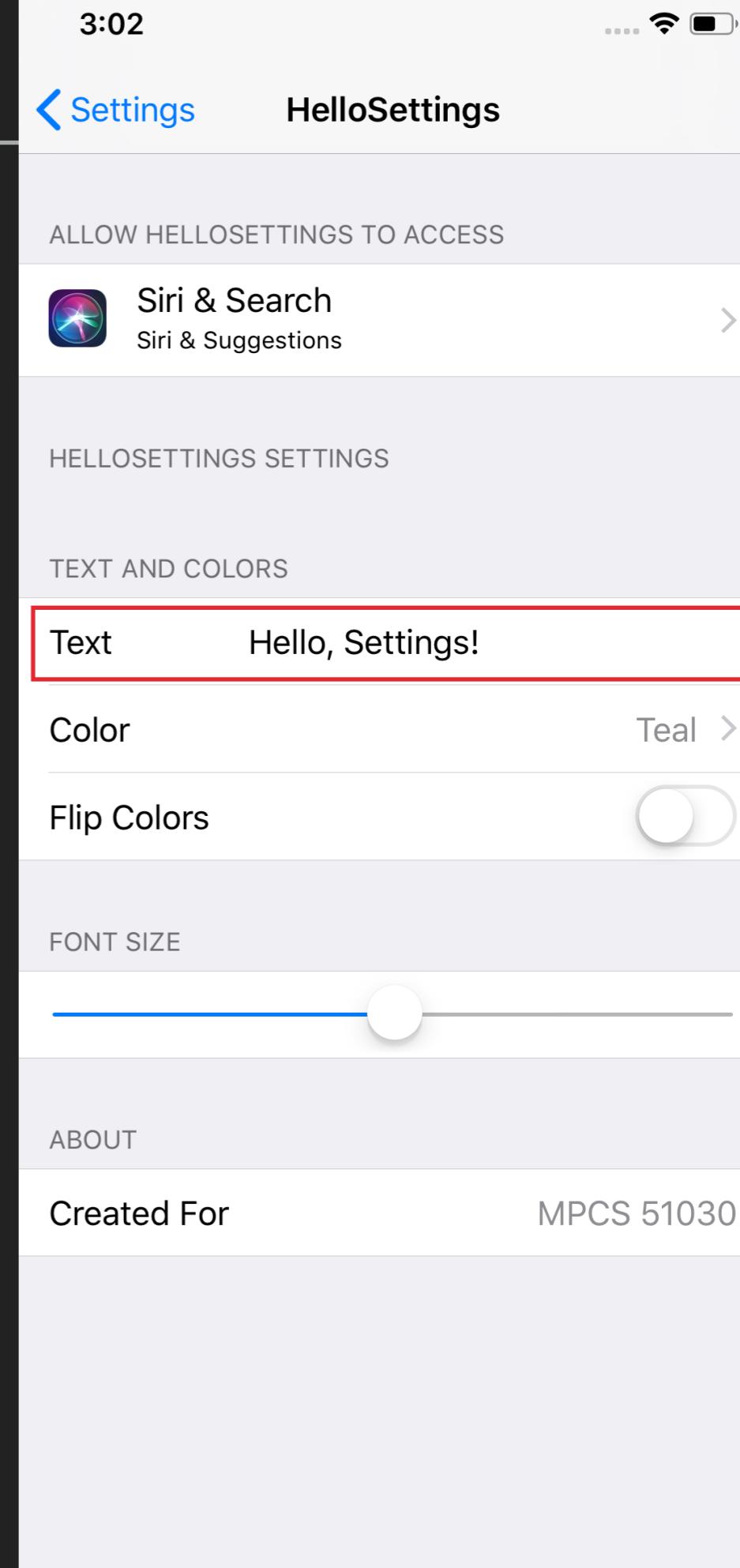
ABOUT

Created For MPCS 51030

SETTINGS BUNDLE

EDITING YOUR SETTINGS BUNDLE

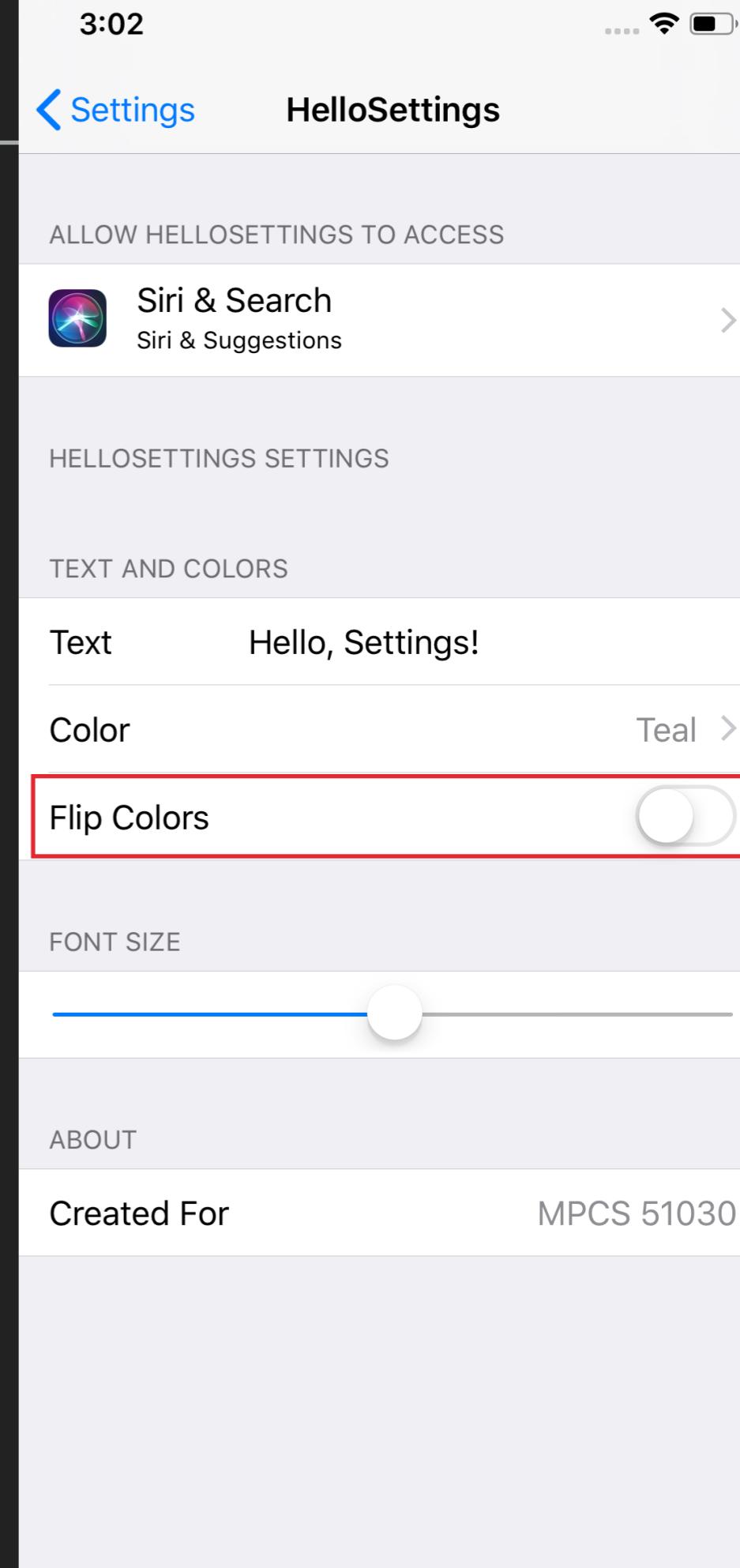
- ▶ Under Preference Items, you can add:
 - ▶ Text Fields (editable by the user)



SETTINGS BUNDLE

EDITING YOUR SETTINGS BUNDLE

- ▶ Under Preference Items, you can add:
 - ▶ Toggle Switches



SETTINGS BUNDLE

EDITING YOUR SETTINGS BUNDLE

► Under Preference Items, you can add:

► Sliders

3:02



◀ Settings

HelloSettings

ALLOW HELLOSETTINGS TO ACCESS



Siri & Search

Siri & Suggestions



HELLOSETTINGS SETTINGS

TEXT AND COLORS

Text

Hello, Settings!

Color

Teal >

Flip Colors



FONT SIZE



ABOUT

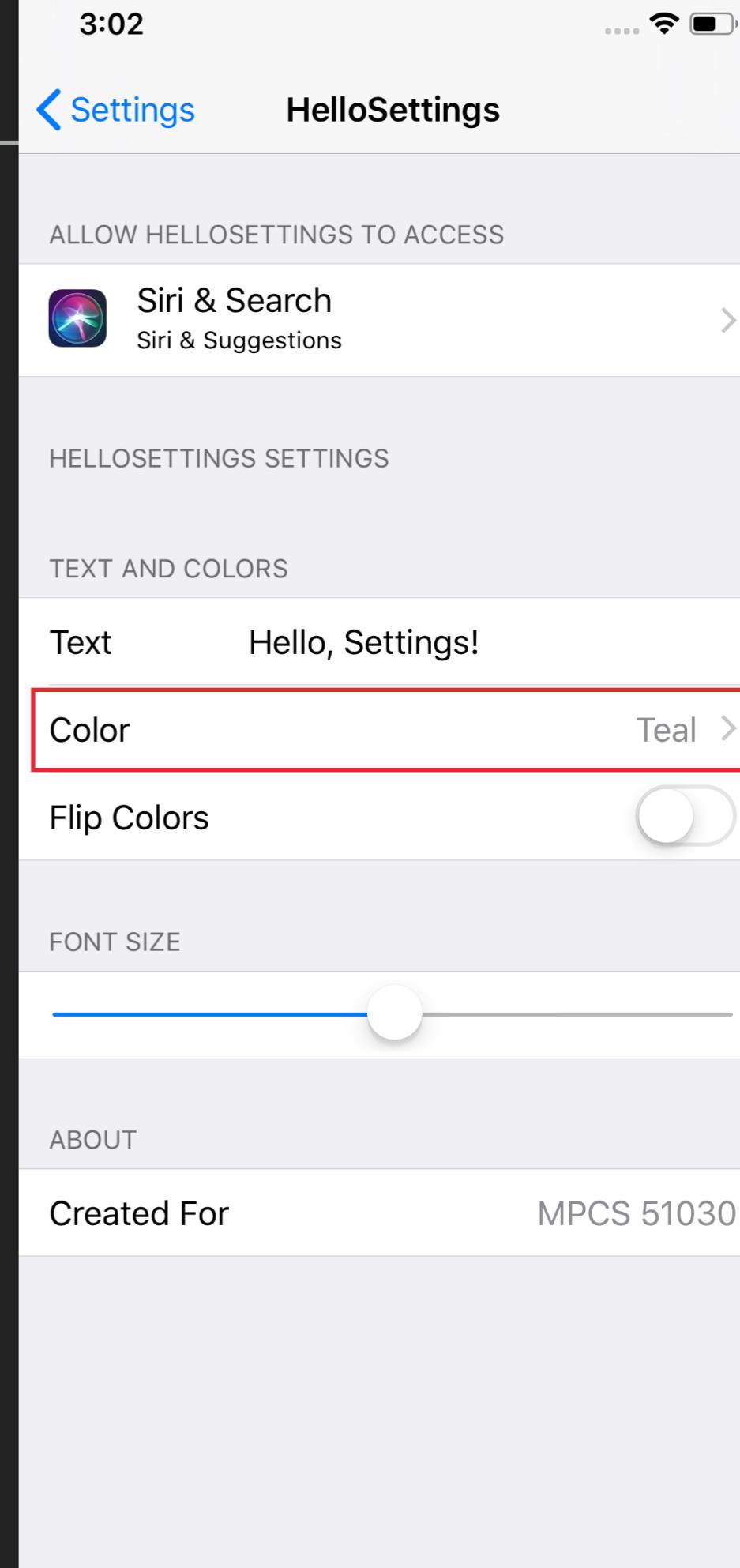
Created For

MPCS 51030

SETTINGS BUNDLE

EDITING YOUR SETTINGS BUNDLE

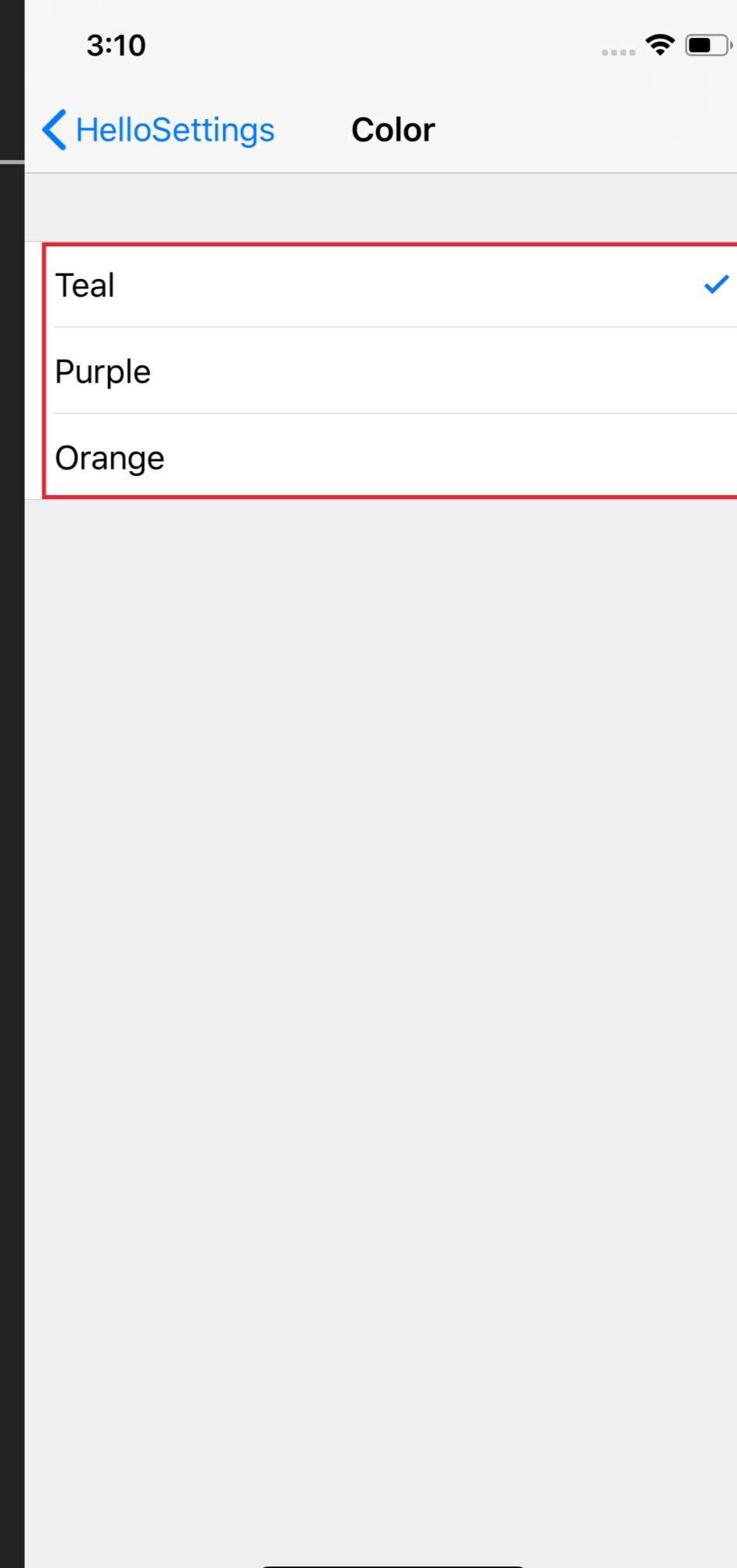
- ▶ Under Preference Items, you can add:
 - ▶ Multi Values



SETTINGS BUNDLE

EDITING YOUR SETTINGS BUNDLE

- ▶ Under Preference Items, you can add:
 - ▶ Multi Values



SETTINGS BUNDLE

EDITING YOUR SETTINGS BUNDLE

- You can edit properties in the Property List view

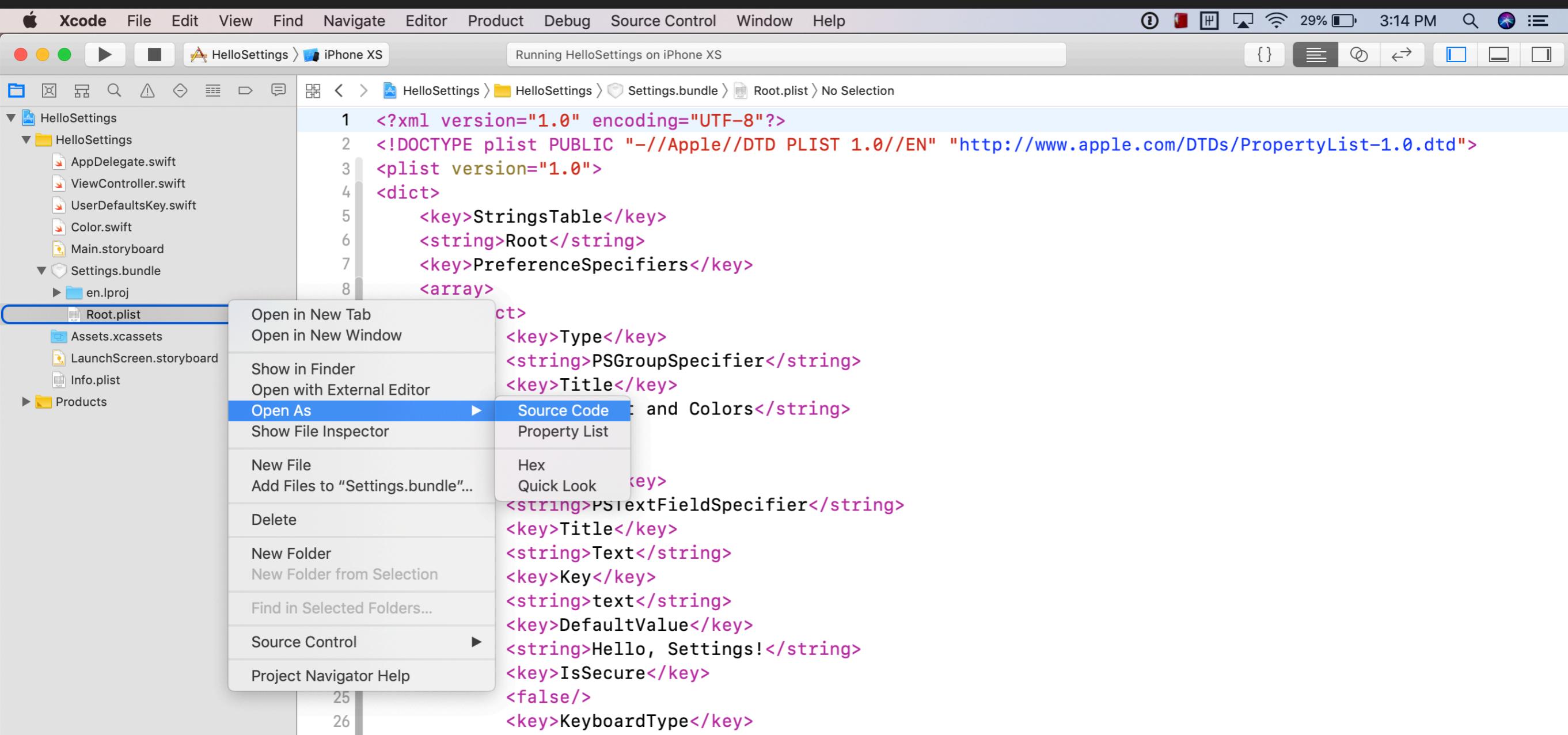
The screenshot shows the Xcode interface with the title bar "Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help". The status bar at the top right shows "Running HelloSettings on iPhone XS" and various system icons. The main window displays the "HelloSettings" project structure on the left, including files like AppDelegate.swift, ViewController.swift, User Defaults Key.swift, Color.swift, Main.storyboard, and a "Settings.bundle" folder containing "en.lproj" and "Root.plist". The "Root.plist" file is open in the center, showing its contents in a table format:

Key	Type	Value
iPhone Settings Schema	Dictionary	(2 items)
Strings Filename	String	Root
Preference Items	Array	(8 items)
Item 0 (Group - Text and Colors)	Dictionary	(2 items)
Type	String	Group
Title	String	Text and Colors
Item 1 (Text Field - Text)	Dictionary	(8 items)
Type	String	Text Field
Title	String	Text
Identifier	String	text
Default Value	String	Hello, Settings!
Text Field Is Secure	Boolean	NO
Keyboard Type	String	Alphabet
Autocapitalization Style	String	None
Autocorrection Style	String	No Autocorrection
Item 2 (Multi Value - Color)	Dictionary	(6 items)
Type	String	Multi Value
Title	String	Color
Identifier	String	colorId
Default Value	String	teal
Values	Array	(3 items)
Item 0	String	teal
Item 1	String	purple
Item 2	String	orange
Titles	Array	(3 items)
Item 0	String	Teal
Item 1	String	Purple
Item 2	String	Orange
Item 3 (Toggle Switch -	Dictionary	(4 items)
Item 4 (Group - Font Size)	Dictionary	(2 items)
Item 5 (Slider)	Dictionary	(7 items)

SETTINGS BUNDLE

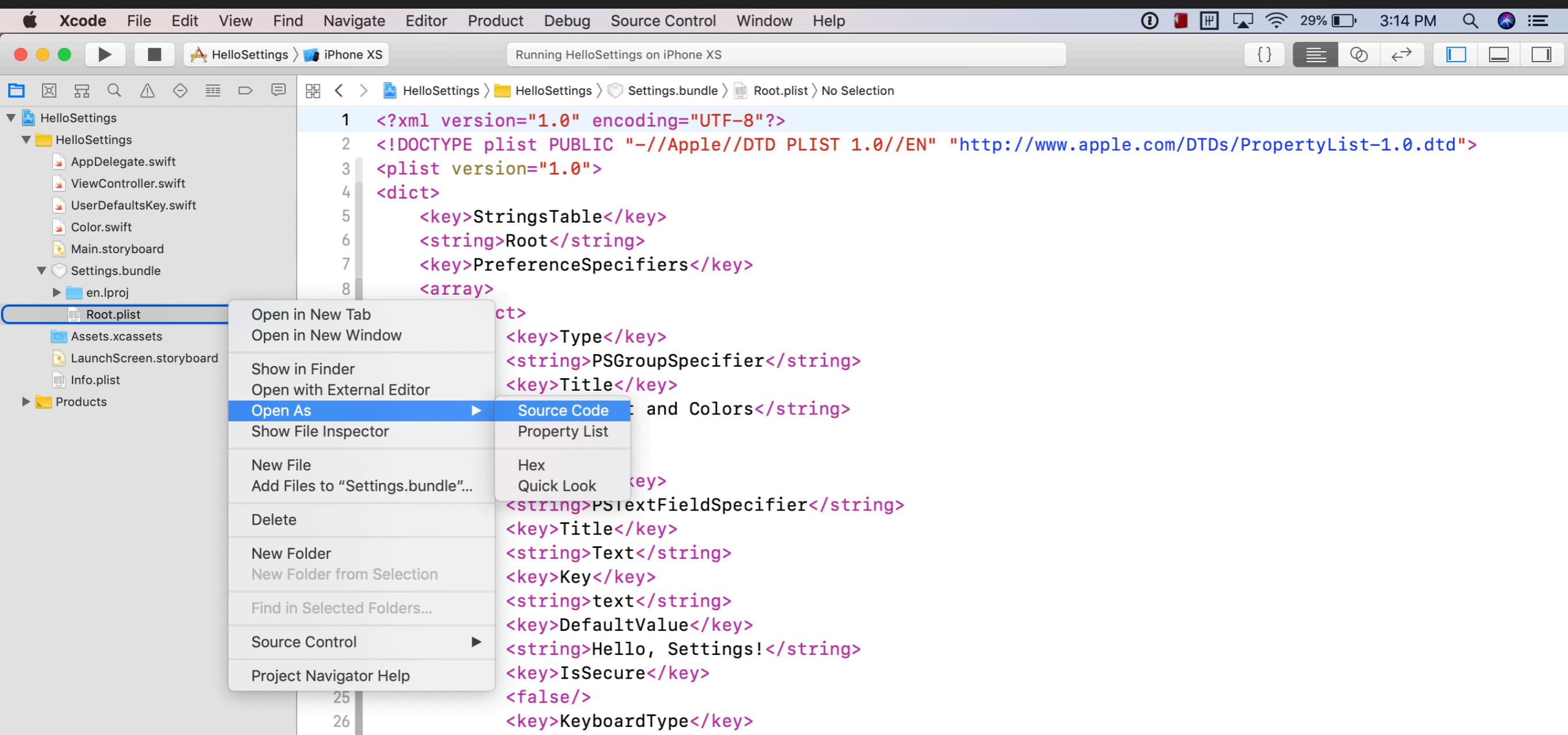
EDITING YOUR SETTINGS BUNDLE

- Or open as Source Code and edit the XML



EDITING YOUR SETTINGS BUNDLE

► Tip! It's easier to rearrange things in XML



ACCESSING SETTINGS IN CODE

- ▶ Access your properties through UserDefaults

```
9 import UIKit
10
11 class ViewController: UIViewController {
12
13     @IBOutlet weak var label: UILabel!
14
15     override func viewDidLoad() {
16         super.viewDidLoad()
17
18         let defaults = UserDefaults.standard
19
20         let text = defaults.string(forKey: UserDefaultsKey.text.rawValue)
21         let shouldFlipColors = defaults.bool(forKey: UserDefaultsKey.shouldFlipColors.rawValue)
22         let fontSize = defaults.double(forKey: UserDefaultsKey.fontSize.rawValue)
23         let colorId = defaults.string(forKey: UserDefaultsKey.colorId.rawValue)
24
25         label.text = text
26         label.font = UIFont.systemFont(ofSize: CGFloat(fontSize), weight: .medium)
27
28         let selectedColor = Color.from(colorId) ?? Color.teal
```

ACCESSING SETTINGS IN CODE

- ▶ Identifiers set in Root.plist are your keys for accessing the values in UserDefaults

The screenshot shows the Xcode interface with the project "HelloSettings" selected. The main area displays the contents of the "Root.plist" file within the "Settings.bundle". The table view shows the following structure and data:

Key	Type	Value
iPhone Settings Schema	Dictionary	(2 items)
Strings Filename	String	Root
Preference Items	Array	(8 items)
Item 0 (Group - Text and Colors)	Dictionary	(2 items)
Item 1 (Text Field - Text)	Dictionary	(8 items)
Type	String	Text Field
Title	String	Text
Identifier	String	text
Default Value	String	Hello, Settings!
Text Field Is Secure	Boolean	NO
Keyboard Type	String	Alphabet
Autocapitalization Style	String	None
Autocorrection Style	String	No Autocorrection
Item 2 (Multi Value - Color)	Dictionary	(6 items)
Item 3 (Toggle Switch -	Dictionary	(4 items)
Item 4 (Group - Font Size)	Dictionary	(2 items)
Item 5 (Slider)	Dictionary	(7 items)
Item 6 (Group - About)	Dictionary	(2 items)
Item 7 (Title - Created For)	Dictionary	(4 items)

PROBLEM WITH DEFAULT VALUES

- ▶ Root.plist allows you to set default values for editable fields

The screenshot shows the Xcode interface with the title bar "Xcode" and various menu options like File, Edit, View, Find, Navigate, Editor, Product, Debug, Source Control, Window, and Help. The toolbar includes standard icons for running, stopping, and refreshing. The main area displays the file structure of a project named "HelloSettings". The "Root.plist" file is open in the editor, showing its contents:

Key	Type	Value
iPhone Settings Schema	Dictionary	(2 items)
Strings Filename	String	Root
Preference Items	Array	(8 items)
Item 0 (Group - Text and Colors)	Dictionary	(2 items)
Item 1 (Text Field - Text)	Dictionary	(8 items)
Type	String	Text Field
Title	String	Text
Identifier	String	text
Default Value	String	Hello, Settings!
Text Field Is Secure	Boolean	NO
Keyboard Type	String	Alphabet
Autocapitalization Style	String	None
Autocorrection Style	String	No Autocorrection
Item 2 (Multi Value - Color)	Dictionary	(6 items)
Item 3 (Toggle Switch -	Dictionary	(4 items)
Item 4 (Group - Font Size)	Dictionary	(2 items)
Item 5 (Slider)	Dictionary	(7 items)
Item 6 (Group - About)	Dictionary	(2 items)
Item 7 (Title - Created For)	Dictionary	(4 items)

PROBLEM WITH DEFAULT VALUES

- ▶ The default values will be shown in the Settings App automatically
- ▶ However, UserDefaults will NOT be aware of those values

SOLUTION FOR DEFAULT VALUES

- ▶ Register defaults when the app launches

```
11 @UIApplicationMain
12 class AppDelegate: UIResponder, UIApplicationDelegate {
13
14     var window: UIWindow?
15
16     func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions:
17                     [UIApplication.LaunchOptionsKey: Any]?) -> Bool {
18
19         guard UserDefaults.standard.string(forKey: "text") == nil else { return true }
20
21         let initialDefaults: [String: Any] = [
22             UserDefaultsKey.text.rawValue : "Hello, Settings",
23             UserDefaultsKey.shouldFlipColors.rawValue : false,
24             UserDefaultsKey.fontSize.rawValue : 30.0,
25             UserDefaultsKey.colorId.rawValue : "teal"
26         ]
27
28         UserDefaults.standard.register(defaults: initialDefaults)
29         return true
30     }
31 }
```

Check if nullable property already has a value

SOLUTION FOR DEFAULT VALUES

- ▶ Register defaults when the app launches

```
11 @UIApplicationMain
12 class AppDelegate: UIResponder, UIApplicationDelegate {
13
14     var window: UIWindow?
15
16     func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions:
17                     [UIApplication.LaunchOptionsKey: Any]?) -> Bool {
18
19         guard UserDefaults.standard.string(forKey: "text") == nil else { return true }
20
21         let initialDefaults: [String: Any] = [
22             UserDefaultsKey.text.rawValue : "Hello, Settings",
23             UserDefaultsKey.shouldFlipColors.rawValue : false,
24             UserDefaultsKey.fontSize.rawValue : 30.0,
25             UserDefaultsKey.colorId.rawValue : "teal"
26         ]
27
28         UserDefaults.standard.register(defaults: initialDefaults)
29         return true
30     }
31 }
```

If not, register initial defaults with UserDefaults

OBSERVING CHANGES WITH

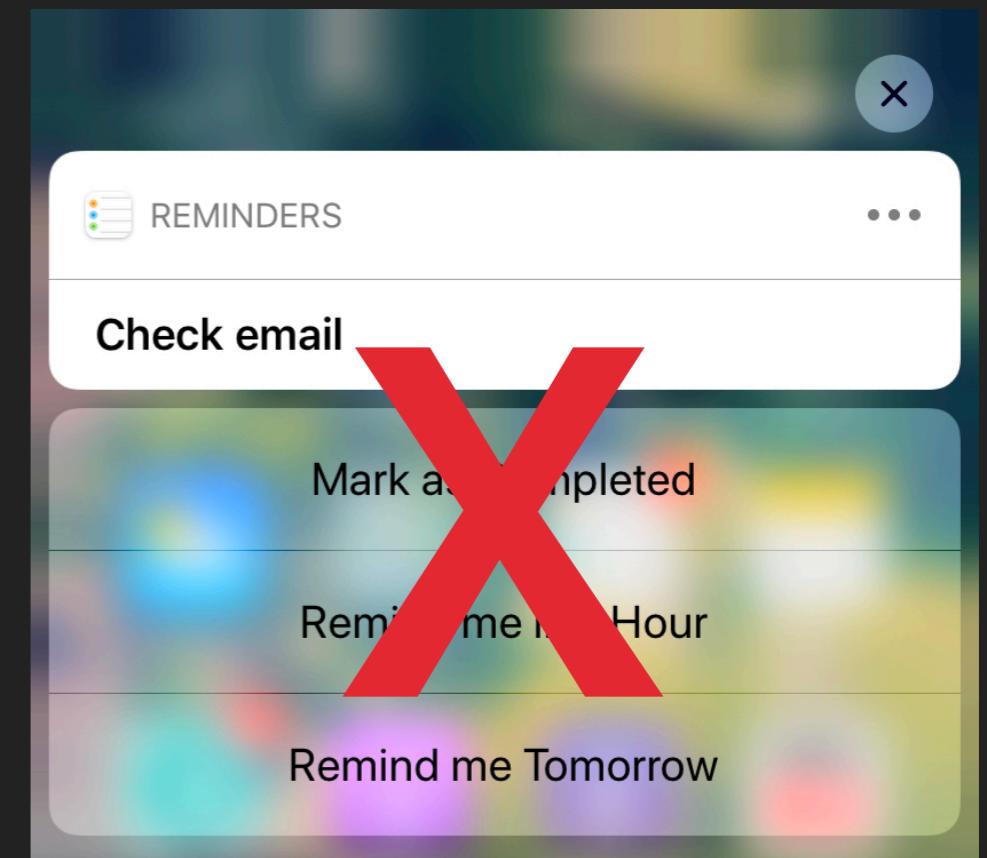
NOTIFICATION CENTER

OBSERVING CHANGES

- ▶ When the user changes a setting, we want to update the app immediately
- ▶ How do we know when a setting has changed?

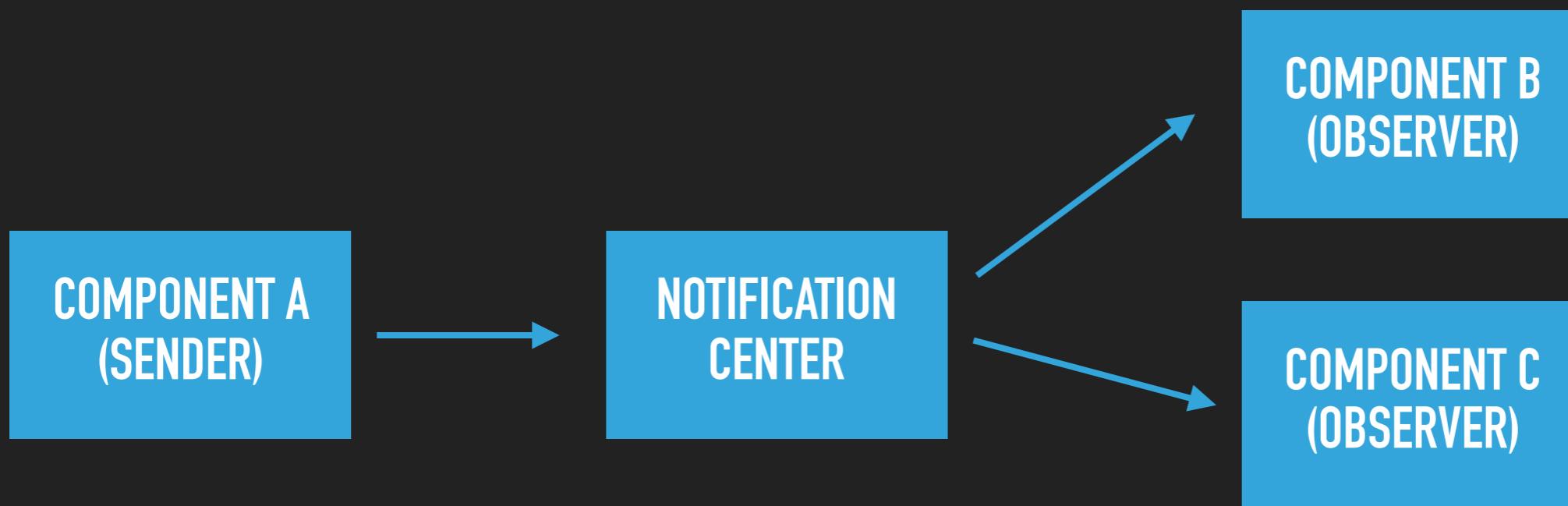
WHAT IS NOTIFICATION CENTER?

- ▶ Mechanism for broadcasting information to different parts of your app
- ▶ Note: This not related to the notifications shown to users



WHAT IS NOTIFICATION CENTER?

- ▶ A sender sends notifications to Notification Center
- ▶ Notification Center dispatches it to registered observers



POSTING A NOTIFICATION

- ▶ Notifications are identified by name
- ▶ Object is the sender of the notification

```
NotificationCenter.default.post(name: Notification.Name("didUpdateUsername"),  
                               object: self)
```

OBSERVING A NOTIFICATION

- ▶ **selector** is the method to be called when the notification is received
- ▶ If **object** is nil, the observer will receive notifications from any sender

```
NotificationCenter.default.addObserver(  
    self,  
    selector: #selector(displayNewUsername),  
    name: Notification.Name("didUpdateUsername"),  
    object: nil)
```

OBSERVING NOTIFICATIONS FROM USER DEFAULTS

```
NotificationCenter.default.addObserver(  
    self,  
    selector: #selector(displaySettings),  
    name: UserDefaults.didChangeNotification,  
    object: nil)
```

OTHER USEFUL NOTIFICATIONS

- ▶ **UIApplication** sends notifications when:
 - ▶ App's state changes (e.g., enters background)
 - ▶ App's orientation changes
 - ▶ User takes a screenshot
 - ▶ ... and many more

OTHER USEFUL NOTIFICATIONS

- ▶ **AVAudioSession** sends notifications when:
 - ▶ Audio is interrupted (e.g., FaceTime call comes in)
 - ▶ Audio route changes (e.g., user unplugs headphones)

OTHER USEFUL NOTIFICATIONS

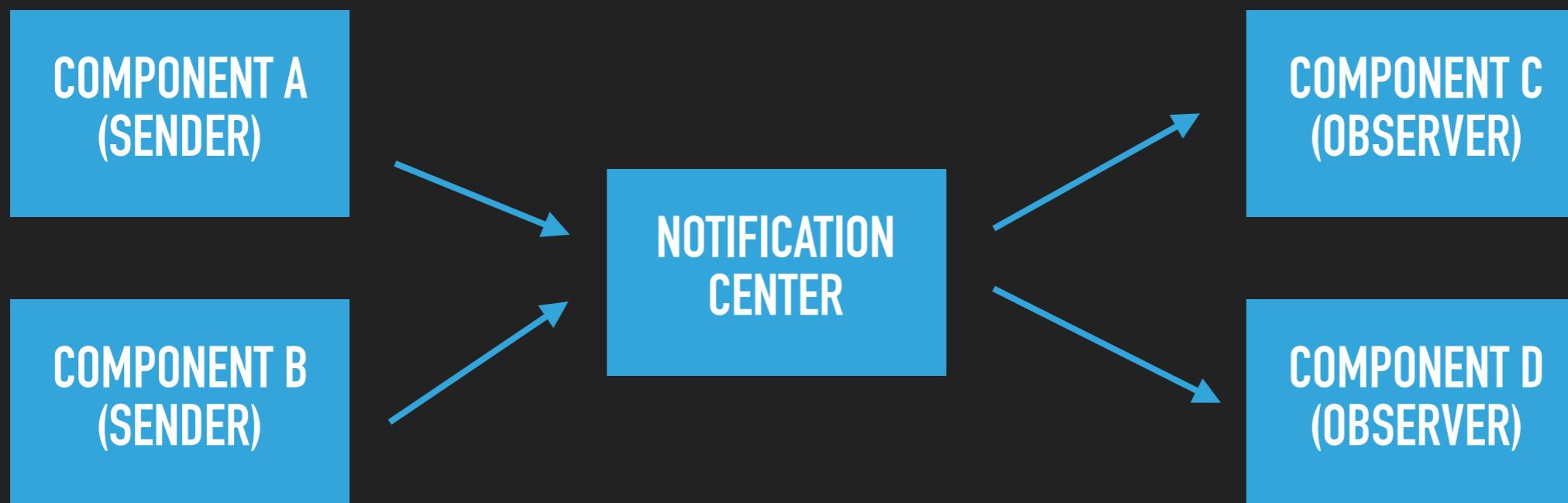
- ▶ **UIResponder** sends notifications:
 - ▶ Just before and after the keyboard is shown and hidden

DOWNSIDE OF NOTIFICATION CENTER

- ▶ One problem with Notification Center: control flow is difficult to follow

DOWNSIDE OF NOTIFICATION CENTER

- ▶ Example: When Component C receives a notification, how do we know where it came from?



RECOMMENDATIONS

- ▶ Use Notification Center only when there is no other good way to be notified of a change
- ▶ Use it primarily to observe events sent by Apple frameworks

RECOMMENDATIONS

- ▶ Before sending your own notifications, consider other ways components can communicate, such as:
 - ▶ Delegates, closures, target-action

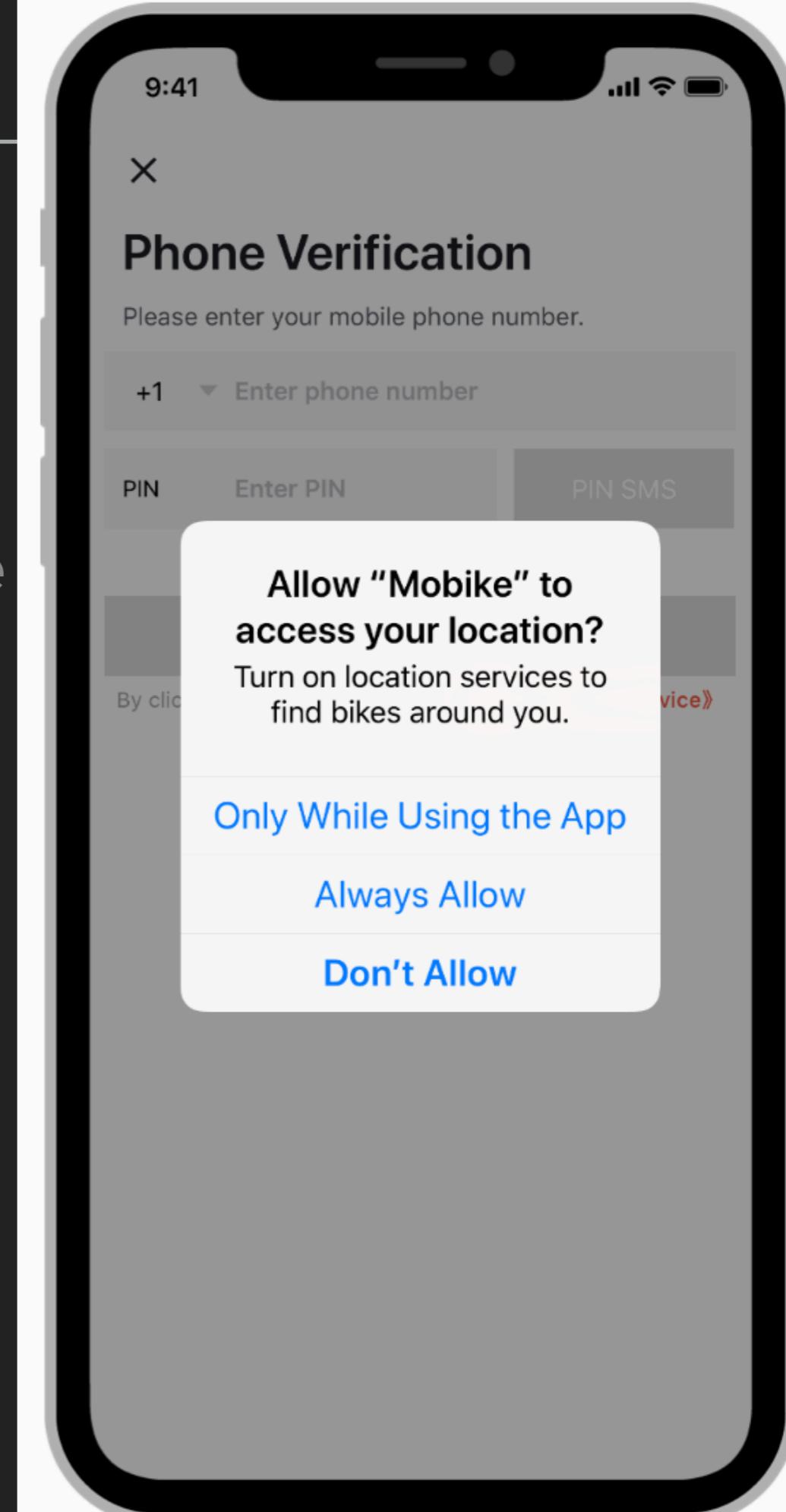
OPENING THE

SETTINGS APP

OPENING THE SETTINGS APP

OPENING THE SETTINGS APP

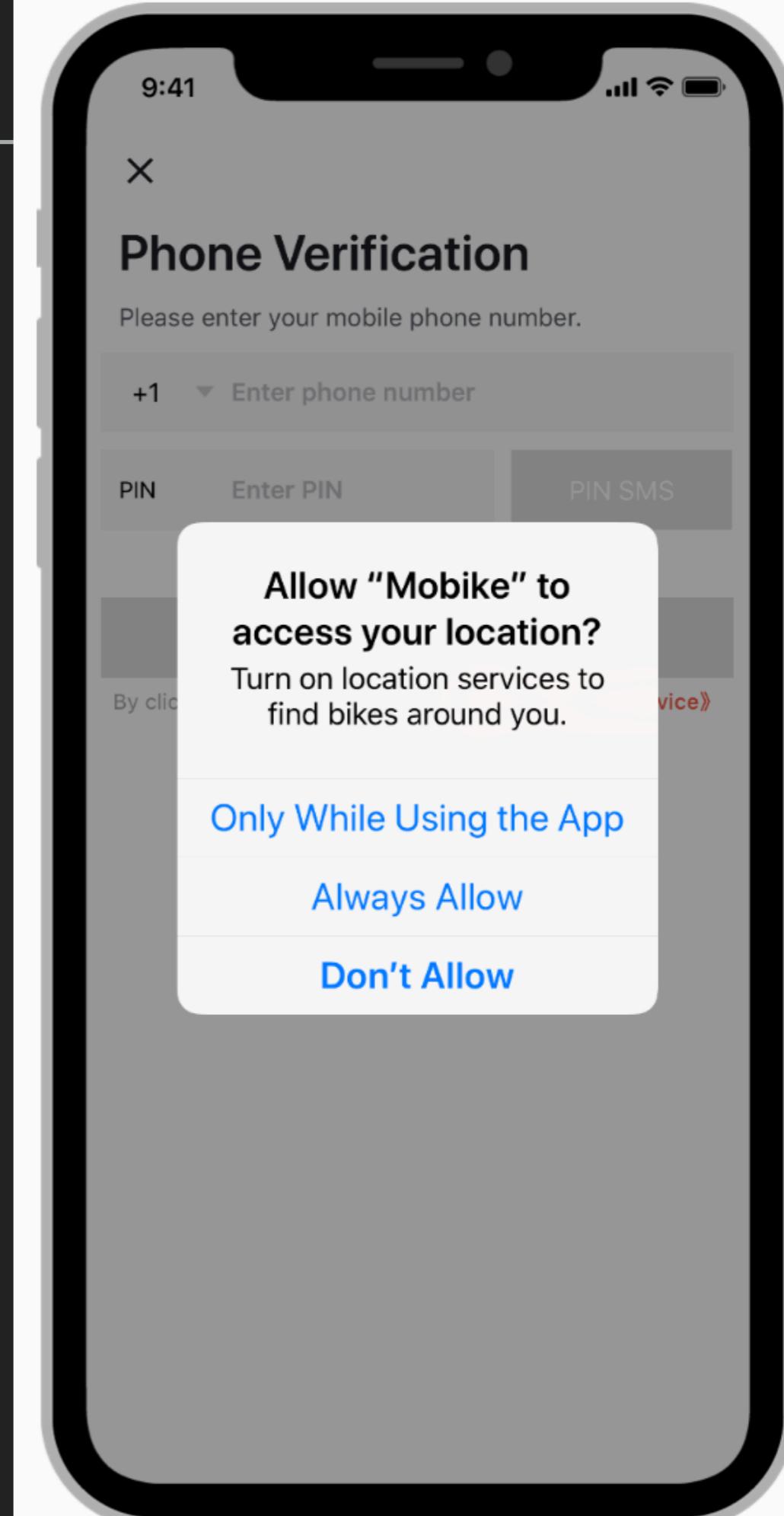
- ▶ You may only show the system prompt requesting a particular type of access (location, camera, contacts, etc.) once



OPENING THE SETTINGS APP

OPENING THE SETTINGS APP

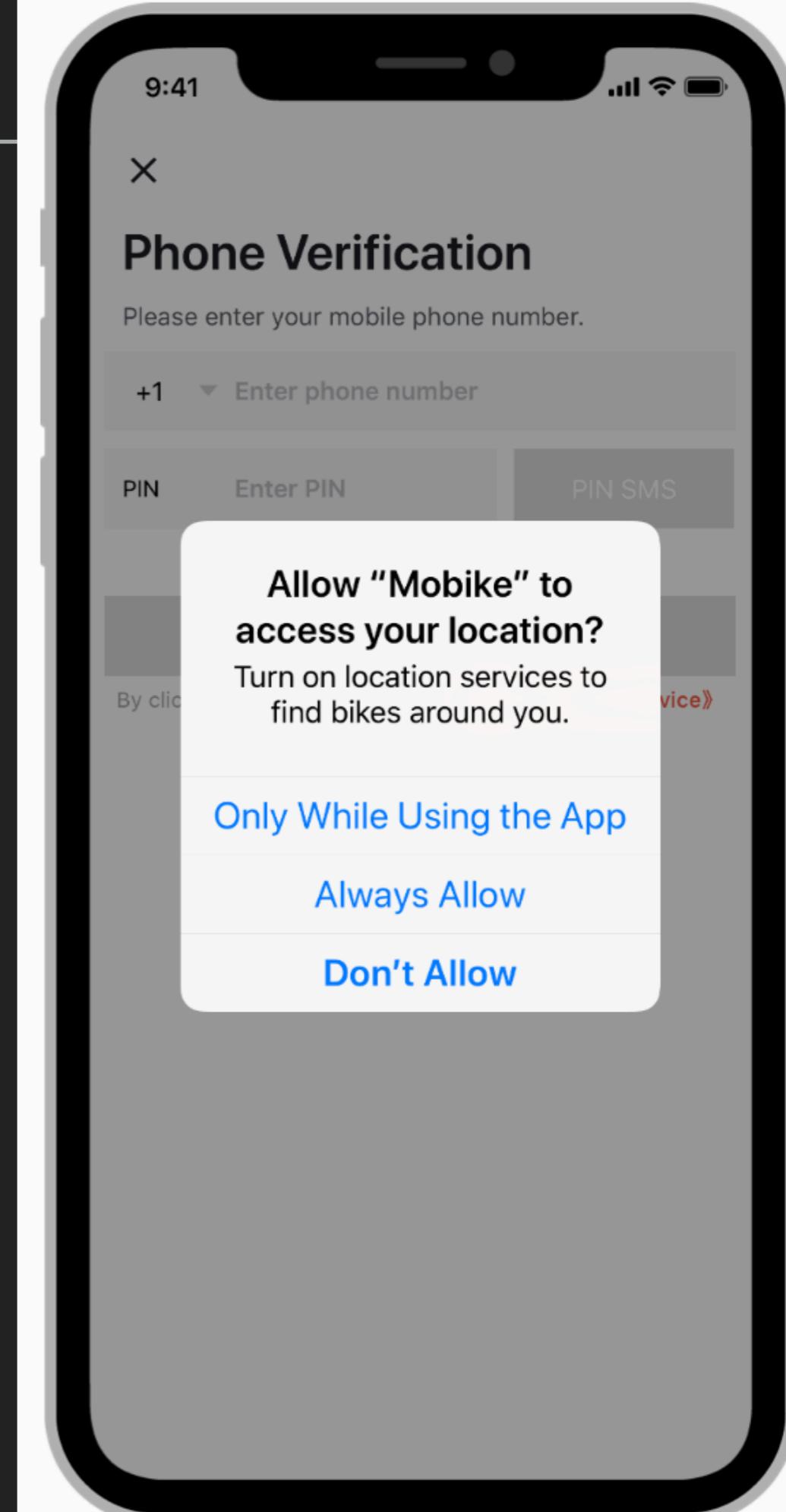
- ▶ If the user denies access initially, the only way for them to grant it later is through the Settings App



OPENING THE SETTINGS APP

OPENING THE SETTINGS APP

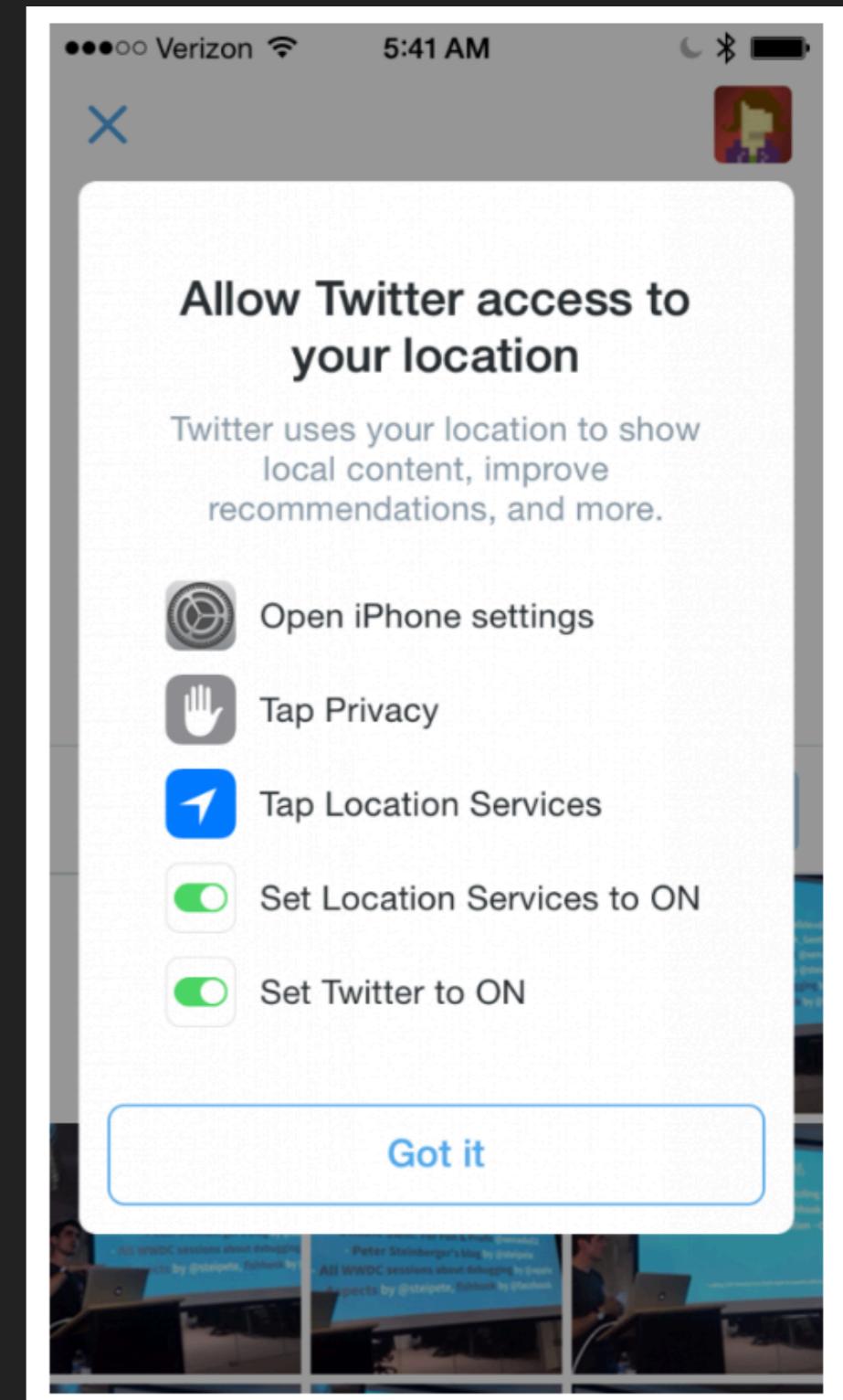
- ▶ Why? Apple doesn't want apps to bug users over and over again.



OPENING THE SETTINGS APP

OLD METHOD

- ▶ Pre iOS 8: No way to automatically send user to the Settings App
 - ▶ Instead, apps provided an elaborate set of instructions



<https://www.natashatherobot.com/ios-taking-the-user-to-settings/>

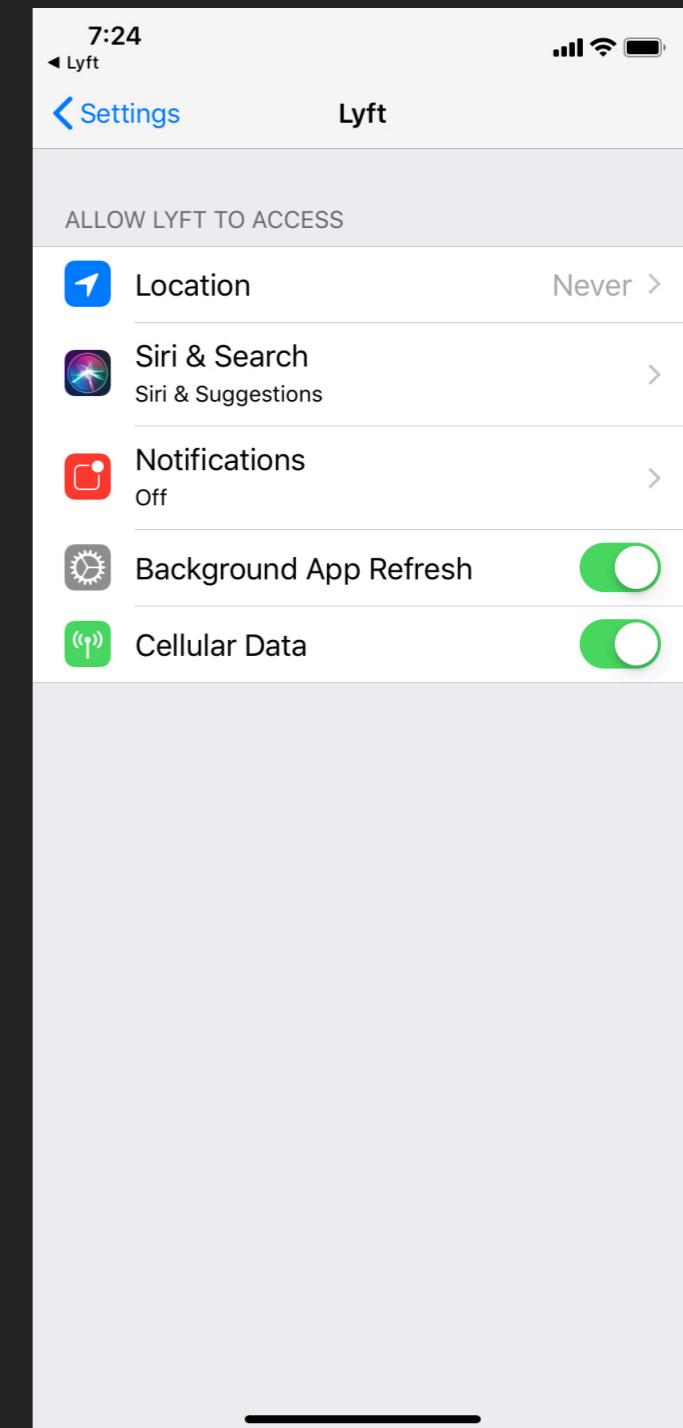
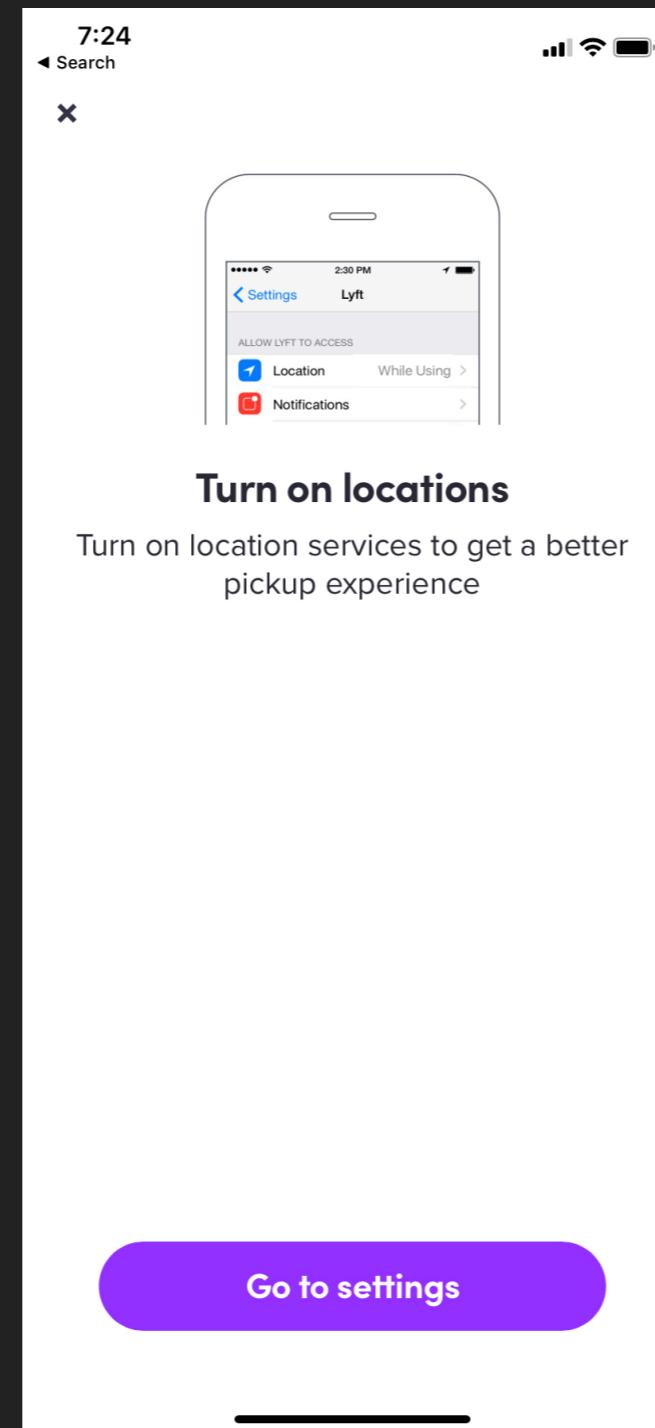
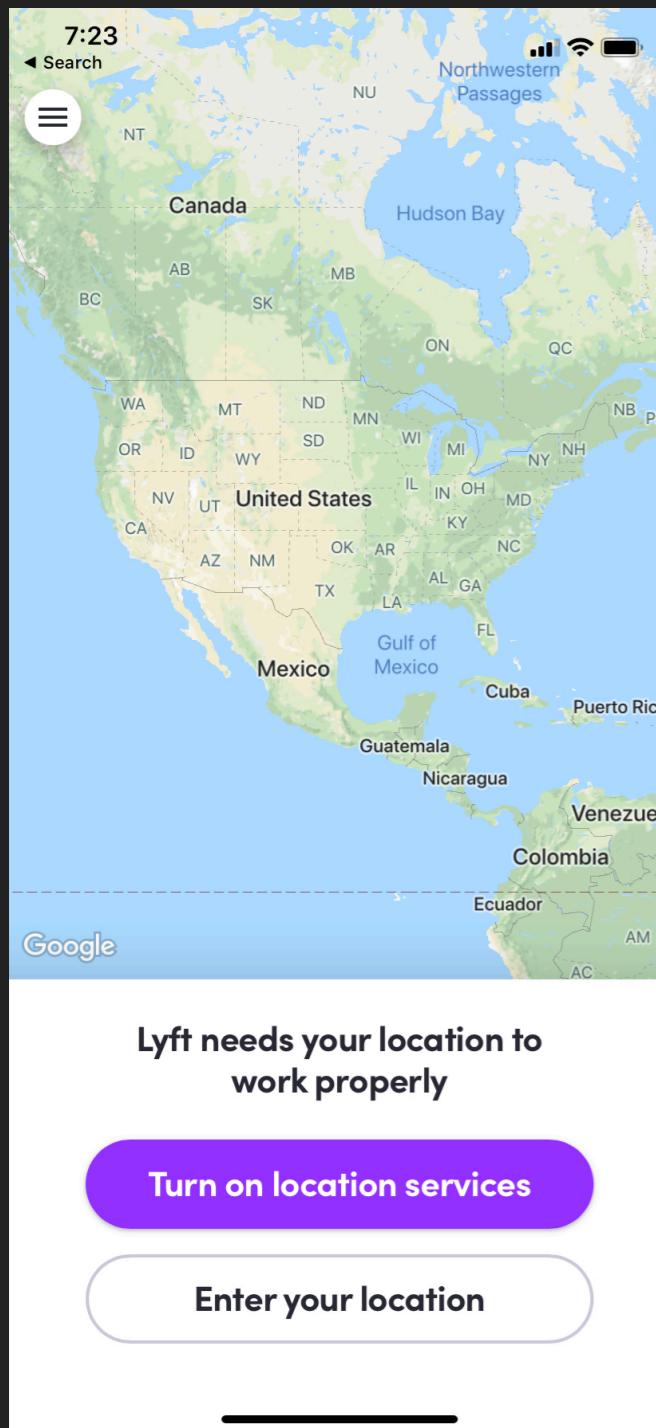
NEW METHOD

- ▶ Deep link the user to your app's specific page in Settings

```
if let settingsUrl = URL(string: UIApplication.openSettingsURLString) {  
    UIApplication.shared.open(settingsUrl)  
}
```

OPENING THE SETTINGS APP

EXAMPLE: LYFT

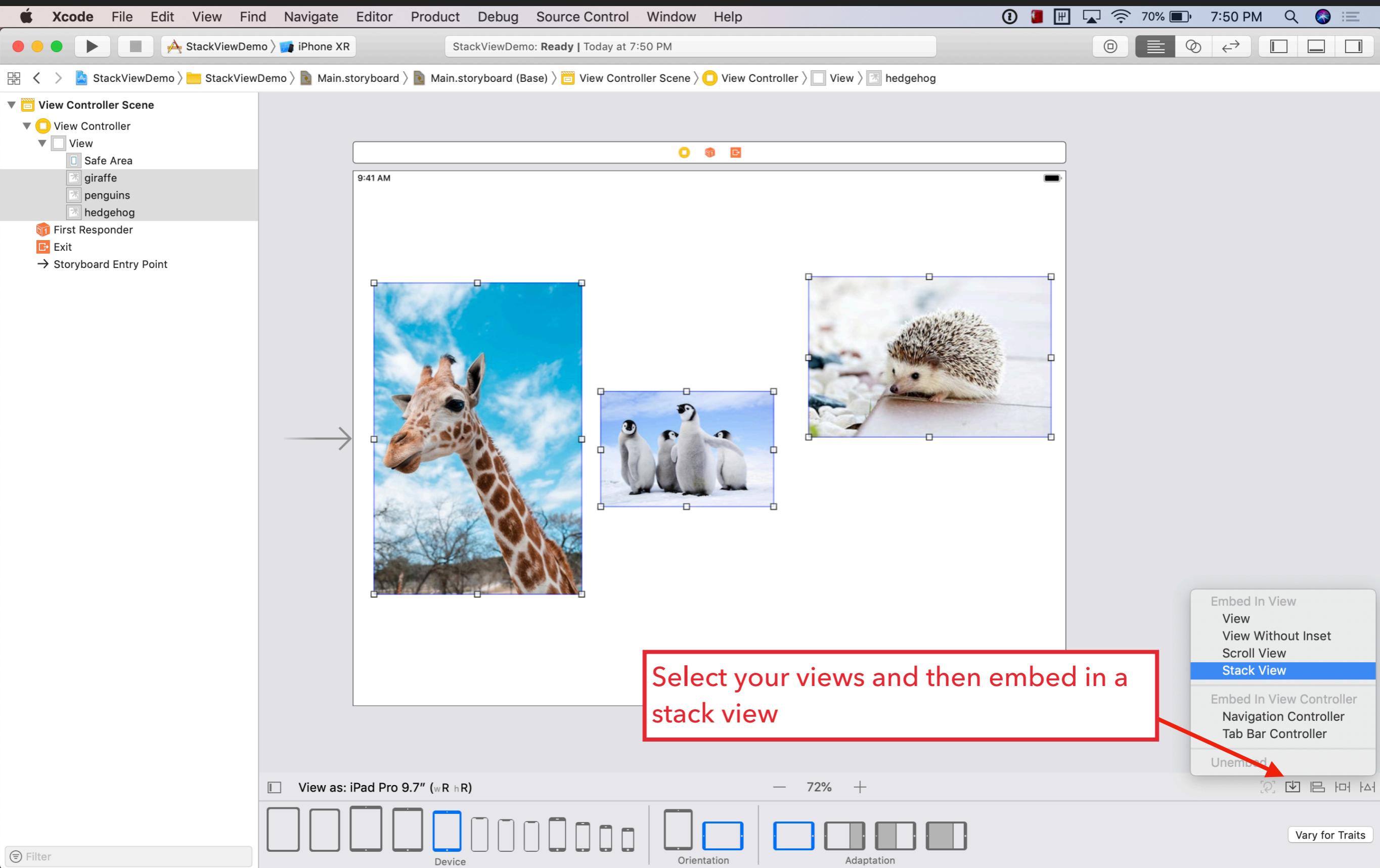


STACK VIEWS

WHAT ARE STACK VIEWS?

- ▶ Containers for laying out views
- ▶ Auto Layout feature introduced in iOS 9
- ▶ Stack views are not rendered on screen (only their subviews are)

STACK VIEWS



STACK VIEWS

Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help

StackViewDemo: Ready | Today at 7:58 PM

StackViewDemo > iPhone XR

StackViewDemo > StackViewDemo > Main.storyboard > Main.storyboard (Base) > View Controller Scene > View Controller > View

View Controller Scene

View Controller

View

Safe Area

Stack View

First Responder

Exit

Storyboard Entry Point

9:41 AM

stack view

Horizontal Stack View - Arranges views linearly.

Vertical Stack View - Arranges views linearly.

You can also drag and drop a stack view from the Object Library

View as: iPad Pro 9.7" (wR hR)

Device Orientation Adaptation

Vary for Traits

Filter

A screenshot of the Xcode interface showing a storyboard for an application named "StackViewDemo". The storyboard is set to run on an "iPhone XR" device. The scene contains three images: a giraffe, a group of penguins, and a hedgehog. On the right side of the screen, the "Object Library" is open, displaying two types of "Stack View": "Horizontal Stack View" and "Vertical Stack View". A callout box with a red border and white text points to the "Horizontal Stack View" entry in the library, with the text "You can also drag and drop a stack view from the Object Library". The bottom of the screen shows the "View as" dropdown set to "iPad Pro 9.7" (wR hR), and the "Device", "Orientation", and "Adaptation" toolbars.

POSITIONING YOUR STACK VIEW

- ▶ Use auto layout constraints to define the position and (optionally) size of your stack view
 - ▶ If you do not define a size, the contents of the stack view will determine its height and width

STACK VIEWS

Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help 8:18 PM 61% 9:41 AM StackViewDemo: Ready | Today at 8:18 PM 1

StackViewDemo StackViewDemo Main.storyboard Main.storyboard (Base) View Controller Scene View Controller View Stack View

View Controller Scene View Controller Scene View Controller View Safe Area Stack View L Pineapple L Strawberry L Kiwi L Peach First Responder Exit

Stack View Axis Horizontal Alignment Fill Distribution Fill Spacing 0 Baseline Relative

View Content Mode Scale To Fill Semantic Unspecified Tag 0 Interaction User Interaction Enabled Multiple Touch Alpha 1 Background Default Tint Default Drawing Opaque Hidden Clears Graphics Context Clip to Bounds Autoresize Subviews Stretching X 0 Y 0 Width 1 Height 1

Add New Alignment Constraints Leading Edges Trailing Edges Top Edges Bottom Edges Horizontal Centers Vertical Centers First Baselines Horizontally in Container Vertically in Container 0 0 Add 2 Constraints

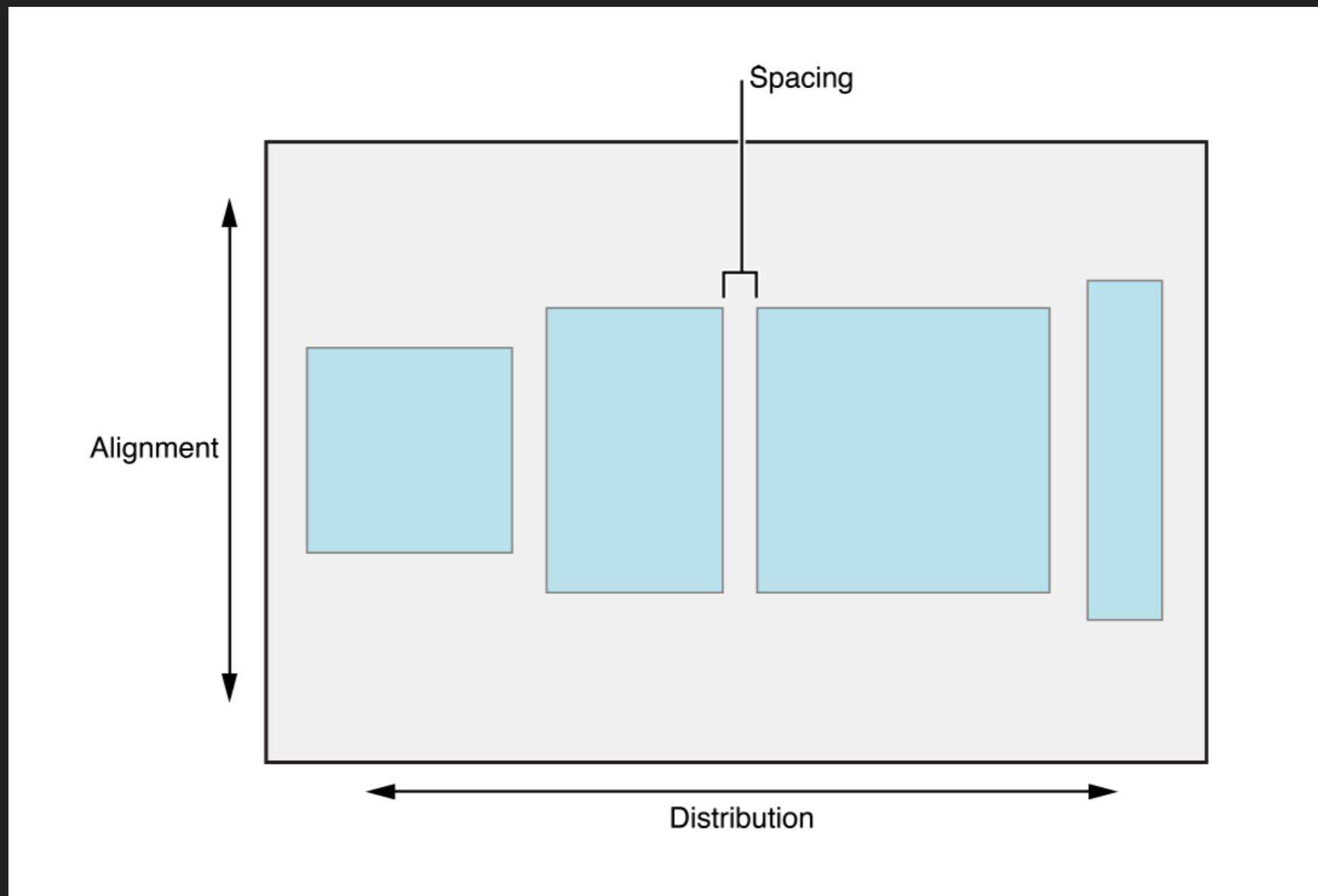
Constraining stack view to the center of the screen

View as: iPad Pro 9.7" (wR hR) 73%

4 MAIN PROPERTIES OF A STACK VIEW

- ▶ Axis
- ▶ Distribution
- ▶ Alignment
- ▶ Spacing

4 MAIN PROPERTIES OF A STACK VIEW



AXIS

- ▶ Either vertical or horizontal

STACK VIEWS

Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help 63% 8:12 PM

StackViewDemo iPhone XR StackViewDemo: Ready | Today at 8:12 PM 1

StackViewDemo StackViewDemo Main.storyboard Main.storyboard (Base) View Controller Scene View Controller View Stack View

View Controller Scene View Controller Scene View Controller View Safe Area Stack View L Pineapple L Strawberry L Kiwi L Peach First Responder Exit

9:41 AM

Pineapple
Strawberry
Kiwi
Peach

Stack View

Axis Vertical

Alignment Fill

Distribution Fill

Spacing 0

Baseline Relative

View

Content Mode Scale To Fill

Semantic Unspecified

Tag 0

User Interaction Enabled

Multiple Touch

Alpha 1

Background Default

Tint Default

Drawing Opaque

Hidden

Clears Graphics Context

Clip to Bounds

Autosize Subviews

Stretching 0 X 0 Y

Width 1 Height 1

Installed

Vertical Axis

Filter View as: iPad Pro 9.7" (wR hR) 73%

STACK VIEWS

Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help 8:14 PM 62% 9:41 AM StackViewDemo: Ready | Today at 8:14 PM 1

StackViewDemo StackViewDemo Main.storyboard Main.storyboard (Base) View Controller Scene View Controller View Stack View

View Controller Scene View Controller Scene View Controller View View Safe Area Stack View L Pineapple L Strawberry L Kiwi L Peach First Responder Exit

Stack View Axis Horizontal Alignment Fill Distribution Fill Spacing 0 Baseline Relative

View Content Mode Scale To Fill Semantic Unspecified Tag 0

Interaction User Interaction Enabled Multiple Touch

Alpha 1

Background Default Tint Default

Drawing Opaque Hidden Clears Graphics Context Clip to Bounds Autoresize Subviews

Stretching 0 0 X 1 1 Y Width Height

Installed

Pineapple Strawberry Kiwi Peach

Horizontal Axis

Filter View as: iPad Pro 9.7" (wR hR) 73% +

The screenshot shows the Xcode interface with a storyboard project named "StackViewDemo". In the storyboard, a single view controller scene is displayed. Inside the view controller, there is a stack view containing four labels: "Pineapple", "Strawberry", "Kiwi", and "Peach", each in a different colored box. The stack view's axis is set to horizontal in the Attributes Inspector, which is highlighted with a red box. The labels are arranged horizontally from left to right. The Xcode status bar at the bottom indicates "View as: iPad Pro 9.7" (wR hR) and a zoom level of 73%.

SPACING

- ▶ Distance between views
 - ▶ May be greater than the value you specify depending on the **distribution** property
 - ▶ Can be set to a negative number to create an overlap

STACK VIEWS

Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help 8:27 PM 59% StackViewDemo iPhone XR StackViewDemo: Ready | Today at 8:27 PM 1

StackViewDemo StackViewDemo Main.storyboard Main.storyboard (Base) View Controller Scene View Controller View Stack View < >

View Controller Scene View Controller Scene View Controller View Safe Area Stack View L Pineapple L Strawberry L Kiwi L Peach Constraints First Responder Exit

Stack View Axis Horizontal Alignment Fill Distribution Fill Spacing 20 Baseline Relative

View Content Mode Scale To Fill Semantic Unspecified Tag 0 Interaction User Interaction Enabled Multiple Touch Alpha 1 Background Default Tint Default Drawing Opaque Hidden Clears Graphics Context Clip to Bounds Autoresize Subviews Stretching 0 0 X 1 1 Width Height Installed

9:41 AM

Pineapple Strawberry Kiwi Peach

Spacing changes the distance between each view

Filter View as: iPad Pro 9.7" (wR hR) 73% +

Spacing

DISTRIBUTION

- ▶ Defines the size and position of views along the stack view's axis

STACK VIEWS

Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help 8:38 PM 55% StackViewDemo: Ready | Today at 8:38 PM

StackViewDemo > iPhone XR

StackViewDemo > StackViewDemo > Main.storyboard > Main.storyboard (Base) > View Controller Scene > View Controller > View > Stack View

View Controller Scene

View Controller Scene

View Controller

View

Safe Area

Stack View

L Pineapple

L Strawberry

L Kiwi

L Peach

Constraints

First Responder

Exit

9:41 AM

Pineapple Strawberry Kiwi Peach

All views have equal width

Stack View

- Axis: Horizontal
- Alignment: Fill
- Distribution: Fill Equally (highlighted)
- Spacing: 0
- Baseline Relative

View

- Content Mode: Scale To Fill
- Semantic: Unspecified
- Tag: 0
- Interaction: User Interaction Enabled (checked)
- Multiple Touch
- Alpha: 1
- Background: Default
- Tint: Default
- Drawing: Opaque, Hidden, Clears Graphics Context (checked), Clip to Bounds, Autoresize Subviews (checked)
- Stretching: X: 0, Y: 0, Width: 1, Height: 1
- Installed

Fill Equally Distribution

Filter View as: iPad Pro 9.7" (wR hR) 73% +

STACK VIEWS

The screenshot shows the Xcode interface with the following details:

- Top Bar:** Xcode, File, Edit, View, Find, Navigate, Editor, Product, Debug, Source Control, Window, Help.
- Toolbar:** Standard Xcode toolbar items.
- Project Navigator:** Shows StackViewDemo > StackViewDemo > Main.storyboard > Main.storyboard (Base) > View Controller Scene > View Controller > View > Stack View.
- Document Outline:** Shows View Controller Scene, View Controller, View, Safe Area, Stack View (selected), and subviews: Pineapple, Strawberry, Kiwi, Peach.
- Assistant Editor:** Shows the Stack View configuration in the Attributes Inspector.
- Attributes Inspector (Stack View section):**
 - Axis: Horizontal
 - Alignment: Fill
 - Distribution: Fill Proportionally** (highlighted with a red box)
 - Spacing: 0
 - Baseline Relative: Unchecked
- Attributes Inspector (View section):**
 - Content Mode: Scale To Fill
 - Semantic: Unspecified
 - Tag: 0
 - Interaction: User Interaction Enabled (checked), Multiple Touch: Unchecked
 - Alpha: 1
 - Background: Default
 - Tint: Default
 - Drawing: Opaque: Unchecked, Hidden: Unchecked, Clears Graphics Context: Checked, Clip to Bounds: Unchecked, Autoresize Subviews: Checked
 - Stretching: X: 0, Y: 0, Width: 1, Height: 1
 - Installed: Checked
- Preview View:** Displays a stack view containing four views labeled Pineapple, Strawberry, Kiwi, and Peach, each with a different colored background (yellow, pink, green, orange).
- Text Overlay:** A red-bordered box contains the text "Views' widths are relative to their contents".
- Bottom Bar:** Filter, View as: iPad Pro 9.7" (wR hR), 73%, +, and other standard Xcode navigation icons.

Fill Proportionally Distribution

STACK VIEWS

The screenshot shows the Xcode interface with a project named "StackViewDemo" for an iPhone XR. The storyboard scene is "View Controller Scene". A stack view containing four arranged views (Pineapple, Strawberry, Kiwi, Peach) is selected. The "Distribution" setting in the Attributes Inspector is highlighted with a red box and set to "Fill Proportionally". A callout bubble with a red border contains the text: "Even if overall width of stack view grows, arranged views maintain their original proportions to each other".

Stack View

- + Axis Horizontal
- + Alignment Fill
- + Distribution **Fill Proportionally**
- + Spacing 0
- + Baseline Relative

View

- Content Mode Scale To Fill
- Semantic Unspecified
- Tag 0
- Interaction User Interaction Enabled Multiple Touch
- Alpha 1
- Background Default
- Tint Default
- Drawing Opaque Hidden Clears Graphics Context Clip to Bounds Autoresize Subviews
- Stretching 0 0 X 1 Width Y 1 Height
- + Installed

Fill Proportionally Distribution

Even if overall width of stack view grows, arranged views maintain their original proportions to each other

STACK VIEWS

Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help 8:45 PM 53%

StackViewDemo iPhone XR StackViewDemo: Ready | Today at 8:45 PM 1

StackViewDemo StackViewDemo Main.storyboard Main.storyboard (Base) View Controller Scene View Controller View Stack View

View Controller Scene View Controller Scene View Controller View Safe Area Stack View L Pineapple L Strawberry L Kiwi L Peach Constraints First Responder Exit

Stack View Axis Horizontal Alignment Fill Distribution Equal Spacing Spacing 0 Baseline Relative

View Content Mode Scale To Fill Semantic Unspecified Tag 0 User Interaction Enabled Multiple Touch Alpha 1 Background Default Tint Default Drawing Opaque Hidden Clears Graphics Context Clip to Bounds Autoresize Subviews Stretching 0 0 X 1 1 Width Height Installed

9:41 AM

Pineapple Strawberry Kiwi Peach

Space between each view is the same

Equal Spacing Distribution

Filter View as: iPad Pro 9.7" (wR hR) 73% +

STACK VIEWS

Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help 8:48 PM 52% 9:41 AM StackViewDemo: Ready | Today at 8:48 PM 1

StackViewDemo StackViewDemo Main.storyboard Main.storyboard (Base) View Controller Scene View Controller View Stack View

View Controller Scene View Controller Scene View Controller View Safe Area Stack View L Pineapple L Strawberry L Kiwi L Peach Constraints First Responder Exit

Stack View Axis Horizontal Alignment Fill Distribution Equal Centering Spacing 0 Baseline Relative

View Content Mode Scale To Fill Semantic Unspecified Tag 0 Interaction User Interaction Enabled Multiple Touch Alpha 1 Background Default Tint Default Drawing Opaque Hidden Clears Graphics Context Clip to Bounds AutoresizeSubviews Stretching 0 0 X 1 1 Y Width Height Installed

Pineapple Strawberry Kiwi Peach

Space from the center of one view to the center of the next is the same

Equal Centering Distribution

Filter View as: iPad Pro 9.7" (wR hR) 73% +

The screenshot shows the Xcode interface with a storyboard project named "StackViewDemo". In the storyboard, a stack view contains four subviews labeled "Pineapple", "Strawberry", "Kiwi", and "Peach", each with a different colored background. The "Distribution" setting in the Attributes Inspector is set to "Equal Centering", which is highlighted with a red border. A red callout box points to the horizontal space between the centers of the views, explaining that the distance from the center of one view to the center of the next is equal. The bottom section of the image features a large blue title "Equal Centering Distribution".

STACK VIEWS

The screenshot shows the Xcode interface with a project named "StackViewDemo" for an iPhone XR. The storyboard scene is "View Controller Scene". A stack view containing four labels ("Pineapple", "Strawberry", "Kiwi", "Peach") is displayed. The stack view's distribution is set to "Fill" in the Attributes Inspector, which is highlighted with a red box. A callout bubble with a red border contains the text: "Stack view fills available space, stretching and compressing views as needed".

Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help

StackViewDemo: Ready | Today at 8:52 PM

Stack View Demo

View Controller Scene

View Controller Scene

View Controller

View

Safe Area

Stack View

L Pineapple

L Strawberry

L Kiwi

L Peach

Constraints

First Responder

Exit

Pineapple

Strawberry

Kiwi

Peach

Stack view fills available space,
stretching and compressing
views as needed

Fill Distribution

Filter

View as: iPad Pro 9.7" (wR hR)

— 73% +

Q Filter

STACK VIEWS

The screenshot shows the Xcode interface with the project "StackViewDemo" open. The storyboard scene "View Controller Scene" is selected. A stack view containing four views ("Pineapple", "Strawberry", "Kiwi", and "Peach") is displayed. The "Pineapple" view is highlighted with a yellow background. The "Strawberry" view is pink, "Kiwi" is green, and "Peach" is orange. A red callout box with a white border and black text is overlaid on the bottom left of the stack view, containing the text: "When there is extra space, the view with the lowest hugging priority is stretched". The right side of the screen shows the Attribute Inspector for the selected "Pineapple" view, with the "Content Hugging Priority" section highlighted by a red border. The horizontal priority is set to 1000 and the vertical to 251.

Content Hugging Priority

- Horizontal: 1000
- Vertical: 251

Content Compression Resistance Priority

- Horizontal: 750
- Vertical: 750

Intrinsic Size: Default (System Defined)

Ambiguity: Always Verify

Fill Distribution

STACK VIEWS

Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help 8:59 PM 49% StackViewDemo iPhone XR StackViewDemo: Ready | Today at 8:59 PM 1

StackViewDemo StackViewDemo Main.storyboard Main.storyboard (Base) View Controller Scene View Controller View Stack View Pineapple

View Controller Scene View Controller Scene View Controller View Safe Area Stack View Pineapple Strawberry Kiwi Peach Constraints First Responder Exit

Label
Desired Width Automatic Explicit

View
Show Frame Rectangle X 0 Y 0 Width 159 Height 48

Arrange Position View

Layout Margins Default
+ Preserve Superview Margins
+ Follow Readable Width
+ Safe Area Relative Margins
Safe Area Layout Guide

Constraints
All This Size Class
This view has no constraints

Content Hugging Priority
Horizontal 1000 Vertical 251

Content Compression Resistance Priority
Horizontal 250 Vertical 750

Intrinsic Size Default (System Defined)
Ambiguity Always Verify

When there is not enough space, the view with the lowest compression resistance priority is compressed

Fill Distribution

Filter View as: iPad Pro 9.7" (wR hR) 73%

The screenshot shows the Xcode interface with a storyboard scene. A stack view contains four views: Pineapple (yellow), Strawberry (pink), Kiwi (green), and Peach (orange). A callout box highlights the 'Content Compression Resistance Priority' settings for the views. The strawberry view has the lowest priority (250), while the others have higher priorities (750). The strawberry view is compressed when there's not enough space.

CONTENT HUGGING PRIORITY

- ▶ Set to an integer between 1 and 1000

Extra space allowed



Low content hugging priority

High content hugging priority

No extra space



CONTENT COMPRESSION RESISTANCE PRIORITY

- ▶ Set to an integer between 1 and 1000

Truncation allowed



Low content compression resistance priority

High content compression resistance priority

No truncation



ALIGNMENT

- ▶ Defines the size and position of views perpendicular to the stack view's axis

STACK VIEWS

Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help 45% 9:15 PM

StackViewDemo > iPhone XR StackViewDemo: Ready | Today at 9:15 PM 1

Stack View Scene View Controller Scene View Controller Scene

View Controller Stack View Stack View

giraffe penguins hedgehog First Responder Exit Storyboard Entry Point

View Controller Stack View Pineapple Strawberry Kiwi Peach Constraints First Responder Exit

Stack View

Axis Horizontal Alignment Top Distribution Fill Spacing 36 Baseline Relative

View

Content Mode Scale To Fill Semantic Unspecified Tag 0

Interaction User Interaction Enabled Multiple Touch

Alpha 1

Background Default Tint Default

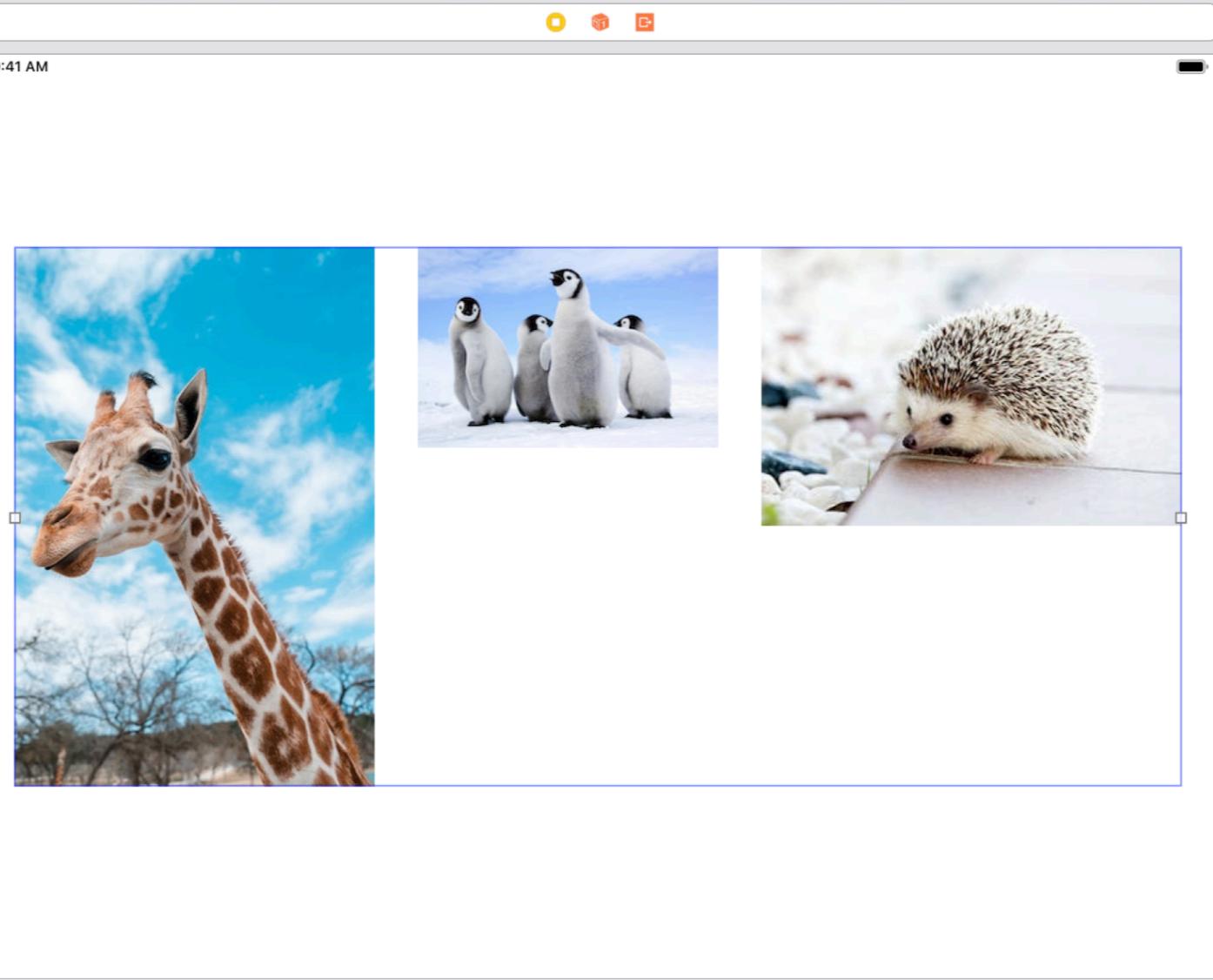
Drawing Opaque Hidden Clears Graphics Context Clip to Bounds Autoresize Subviews

Stretching 0 0 X 1 1 Y Width Height

Installed

Top Alignment

Filter View as: iPad Pro 9.7" (wR hR) 73% +



STACK VIEWS

Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help 45% 9:16 PM

StackViewDemo > iPhone XR StackViewDemo: Ready | Today at 9:16 PM 1

Stack View Scene View Controller Scene View Controller Scene

View Controller Scene View Controller View Safe Area Stack View giraffe penguins hedgehog First Responder Exit Storyboard Entry Point

View Controller Scene View Controller View Safe Area Stack View L Pineapple L Strawberry L Kiwi L Peach Constraints First Responder Exit

Stack View

- + Axis Horizontal
- + Alignment Center **Center**
- + Distribution Fill
- + Spacing 36
- + Baseline Relative

View

- Content Mode Scale To Fill
- Semantic Unspecified
- Tag 0
- Interaction User Interaction Enabled Multiple Touch
- Alpha 1
- Background Default
- Tint Default
- Drawing Opaque Hidden Clears Graphics Context Clip to Bounds Autoresizes Subviews
- Stretching 0 0 X 1 1 Y Width Height
- + Installed

9:41 AM

Center Alignment

Filter View as: iPad Pro 9.7" (wR hR) — 73% +

STACK VIEWS

Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help 9:17 PM 45% ⚡

StackViewDemo iPhone XR StackViewDemo: Ready | Today at 9:17 PM 1

StackViewDemo StackViewDemo Main.storyboard Main.storyboard (Base) View Controller Scene View Controller View Stack View

View Controller Scene View Controller View Safe Area Stack View giraffe penguins hedgehog First Responder Exit Storyboard Entry Point

View Controller Scene View Controller View Safe Area Stack View Pineapple Strawberry Kiwi Peach Constraints First Responder Exit

Stack View

- + Axis Horizontal
- + Alignment **Bottom**
- + Distribution Fill
- + Spacing 36
- + Baseline Relative

View

- Content Mode Scale To Fill
- Semantic Unspecified
- Tag 0
- Interaction User Interaction Enabled Multiple Touch
- Alpha 1
- Background Default
- Tint Default
- Drawing Opaque Hidden Clears Graphics Context Clip to Bounds Autoresizes Subviews
- Stretching 0 X 0 Y 1 Width 1 Height 1
- + Installed

9:41 AM

Bottom Alignment

Filter View as: iPad Pro 9.7" (wR hR) — 73% +

STACK VIEWS

Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help 44% 9:17 PM

StackViewDemo > iPhone XR StackViewDemo: Ready | Today at 9:17 PM 1

Stack View Scene View Controller Scene View Controller Scene

View Controller Stack View Stack View

giraffe penguins hedgehog

First Responder Exit Storyboard Entry Point

View Controller Scene View Controller Scene

View Controller Stack View

L Pineapple L Strawberry L Kiwi L Peach

First Responder Exit

Stack View

Axis Horizontal

Alignment Fill (highlighted)

Distribution Fill

Spacing 36

Baseline Relative

View

Content Mode Scale To Fill

Semantic Unspecified

Tag 0

Interaction User Interaction Enabled (checked)

Multiple Touch

Alpha 1

Background Default

Tint Default

Drawing Opaque (unchecked)

Hidden (unchecked)

Clears Graphics Context (checked)

Clip to Bounds (unchecked)

Autoresizing Subviews (checked)

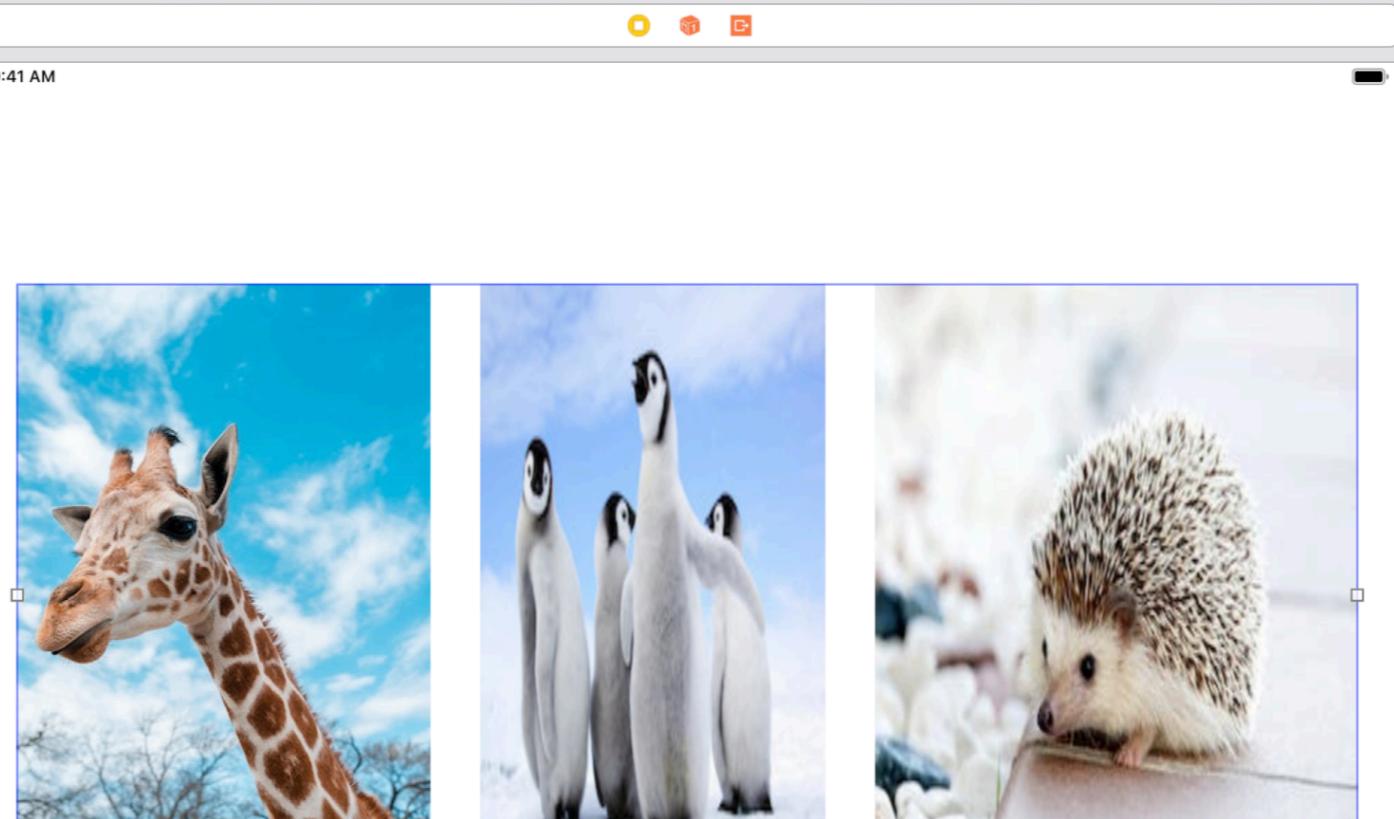
Stretching 0 X 0 Y

Width 1 Height 1

Installed

Fill Alignment

Filter View as: iPad Pro 9.7" (wR hR) 73%



STACK VIEWS

Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help 9:25 PM 42%

StackViewDemo > iPhone XR StackViewDemo: Ready | Today at 9:25 PM 2

Stack View Scene View Controller Scene View Controller Scene View Controller Scene

View Controller View Controller View Controller

Stack View Stack View Stack View

giraffe penguins hedgehog

This is a label with multiple lines Here's another label

Stack View

Axis Horizontal

Alignment First Baseline (highlighted)

Distribution Fill

Spacing 38

Baseline Relative

View

Content Mode Scale To Fill

Semantic Unspecified

Tag 0

Interaction User Interaction Enabled

Multiple Touch

Alpha 1

Background Default

Tint Default

Drawing Opaque

Hidden

Clears Graphics Context

Clip to Bounds

Autoresizing Subviews

Stretching 0 X 0 Y

Width 1 Height 1

Installed

Filter View as: iPad Pro 9.7" (wR hR) 73% +

First Baseline Distribution

STACK VIEWS

Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help 41% 9:25 PM

StackViewDemo > iPhone XR StackViewDemo: Ready | Today at 9:25 PM 2

Stack View Scene View Controller Scene View Controller Scene View Controller Scene

View Controller Scene View Controller View Stack View giraffe penguins hedgehog First Responder Exit Storyboard Entry Point

View Controller Scene View Controller View Stack View L This is a label with multi... L Here's another label Constraints First Responder Exit

View Controller Scene View Controller View Stack View L Pineapple L Strawberry L Kiwi L Peach Constraints First Responder Exit

Stack View

- + Axis Horizontal
- + Alignment Last Baseline **Last Baseline**
- + Distribution Fill
- + Spacing 38
- + Baseline Relative

View

- Content Mode Scale To Fill
- Semantic Unspecified
- Tag 0
- Interaction User Interaction Enabled Multiple Touch
- Alpha 1
- Background Default
- Tint Default
- Drawing Opaque Hidden Clears Graphics Context Clip to Bounds AutoresizesSubviews
- Stretching 0 X 0 Y 1 Width 1 Height 1
- + Installed

This is a label with multiple lines

Here's another label

Last Baseline Distribution

Filter View as: iPad Pro 9.7" (wR hR) — 73% +

STACK VIEWS

STACK VIEW MAGIC

- ▶ When you hide one view, the others will redistribute themselves accordingly
- ▶ You can even animate this change

Reminders

10:16



To Do

3

Edit

Submit final project proposal

Today, 11:00 PM

Think of an awesome app idea. It should be really cool.

!!! Finish assignment 5

Work on case study

Make slides.

+